64 PLUS 4 & AMIGA



KWIECIEŃ 1991 ISSN 0867-3918 INDEKS 377112 CENA 5000 zł

MIESIĘCZNIK UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE

W NUMERZE:

- Sampling
- Space Ace
- Reset

BUK

W numerze :

W następnym numerze :

- Stacje wysokiej gęstości
- LEMINGS
- Korektor kodu maszynowego C-64
- Własna dyskietka na Amigę

OGŁOSZENIA

KOMPUTEROWA FIRMA USŁUGOWA "TREND" - COMMODORE AMIGA: oprogramowanie, literatura. Informacja: dyskietka lub koperta + znaczek. Kontakt: Rafał Wierzbicki, ul. Rogowska 86/10, 54-440 Wrocław.

C-16 - wymiana oprogramowania. M. Bochenek, 39-200 Dębica, ul. Łąkowa 14.

SUPERTEST. I nagroda 500.000zł. Info - k+z+4000zł. Szyduk Przemysław, ul. T. Kościuszki 71/14, 63-200 Jarocin.

„THE BLOD GREMLINS" nawiąże kontakt z Amigowcami. Wójcicki Kamil, ul. Nowowolska 60, 95-200 Pabianice.

Wymienię oprogramowanie na Amigę. Andrzej Bogdan, ul. Okrzei 7/16, 76-100 Sławno.

SPRZEDAM AMIGĘ 500 I MONITOR KOLOROWY. St. Wojciechowski, Kalisz, tel 533-30.

Rozrzerzenie do Amigi 500 - 0.5MB - 600000zł i 2MB - 2mln zł. M. Skoneczko, ul Rąbieńska 24, 94-227 Łódź

Sprzedam C-64II, stację dysków, 70 gier, literaturę. Piła, tel. 243-65 - wieczorem.

EXCLUSIVE CLUB
GRY I PROGRAMY NA KAŻDY KOMPUTER
ZA DARMO!!!
INFORMACJE - KOPERTA+ZNACZEK
Artur Makowski
ul. Worcella 34m9
42-200 Częstochowa

KORESPONDENCYJNY KLUB KOMPUTEROWY C-64. Informacje = koperta + znaczek. Ul. Kaliny 25/8, 71-118 Szczecin.

Sprzedam monitory kolorowe Commodore 1801, 1901 - cena: po 2mln zł. Katowice, tel. 544-565.

Uwaga!
MOJA AMIGA
- książka autorska napisana prostym językiem.
Informacje: M. Pampuch, Kraków, tel 0-12-471 761.

Klub Komputerowy Stodoła
AMIGA
- oferuje najlepsze stacje dysków 3,5" i 5,25"
- serwis sprzętu firmy Commodore
- literatura (także 64 plus 4)
- akcesoria itp.

Zapraszamy codziennie, oprócz sobót i niedziel w godzinach 11⁰⁰ - 20⁰⁰
Warszawa ul. Batorego 10
tel. 25-60-31 wew. 35.
Giełdy komputerowe w Stodole, sobota od 10⁰⁰ - 15⁰⁰

64 PLUS 4
miesięcznik nr 4(6) kwiecień 1991
cena 1 egz.: 5000 zł



Wydawca:
ABUK Spółka z o. o.
Redakcja nie ponosi odpowiedzialności za treść ogłoszeń.

Adres redakcji: Redakcja „64 plus 4" 85-166 Bydgoszcz 43 skrytka pocztowa 64
Redagują: Marcin Dudar, Sambor Kuźma, Paweł Sołtysiński, Waldemar Szczygieł (red. nacz.), Robert Turliński, Wojciech Wasilewski.
Okładka: Sławomir Karolczak.
Skład: ABUK
Druk: Z.P. POLRASTER
85-353 Bydgoszcz, ul. Orawska 19

Drodzy Czytelnicy!

Oto oddajemy Wam do rąk kolejny numer naszego pisma. Jego treść - jak nigdy dotąd - wypełniają artykuły napisane w oparciu o propozycje i sugestie zawarte w Waszych listach. W dalszym ciągu zapraszamy Was do współpracy!

Otrzymaliśmy wiele słów krytyki za duże opóźnienia w kolprotażu naszego pisma i dyskietek - jeszcze raz przepraszamy. Zreorganizowaliśmy nasz dział koloprtażu tak, aby wszelkie zamówienia były realizowane terminowo.

Kilku czytelników krytykowało nas za zamieszczanie ogłoszeń osób, które zajmują się handlem oprogramowaniem (nie posiadając podstaw prawnych) - uwagi te dotyczyły ogłoszeń bezpłatnych. Dziękujemy za zwrócenie nam uwagi na tą sprawę. Nie popieramy „piractwa"! Jako bezpłatne przyjmować będziemy tylko te ogłoszenia, które nie naruszają wspomnianych zasad.

Nowa forma graficzna okładki jest krokiem w kierunku podnoszenia atrakcyjności i estetyki miesięcznika. Mamy nadzieję, iż w najbliższym czasie uda nam się powiększyć objętość pisma, co pozwoli na szersze wykorzystanie ilustracji i grafiki wewnątrz numeru (na razie staramy się, jak najwięcej miejsca przeznaczać na artykuły).

Tocząca się „wojna" między zwolennikami komputerów Atari i Commodore przybiera różne formy - od zaciekłości po elementy humorystyczne. My wychodzimy z założenia, że najważniejszy jest człowiek i sposób w jaki wykorzystuje on swoją „maszynę". Uwagi krytyczne na temat Atari, które znajdziecie w dziale nowinek i słowniku slangu, traktujcie więc z przymrużeniem oka.

Redakcja

Z daleka i z bliska

- Z BitNet'u dowiedzieliśmy się o nowej kości dźwiękowej do Amigi. Nosi ona nazwę „Mary" i ma być umieszczana w miejsce Pauli. Firma twierdzi, że Mary jakością dźwięku nie będzie odbiegała od kompaktu. Kość ma mieć możliwość grania na osiem kanałów!

- Na IBM pojawił się Screanstracker który jest odpowiednikiem Amigowskiego Soundtracker'a. Dr. Awesome (jeden z najlepszych muzyków na scenie) stwierdził, że edycja jest niezła, lecz jakość dźwięku jest do kitu.

- W Finlandii odbyła się dyskoteka na której grano wyłącznie muzykę z Amigi (Noisetracker)! Impreza była ponoć sukcesem. Niektórzy z muzyków dostali nawet swój udział z zysków. Z dumą musimy przyznać, że nie była to pierwsza impreza na której bawiono się przy muzyce z Amigi. Redaktorzy „64 PLUS 4" już dwa razy zorganizowali coś podobnego w Szczecinie. Jedyny dochód jaki przyniosły im te dyskoteki pochodził z zakładów typu „O ile się założysz, że nie zgadniesz jaki zespół gra tą muzykę?". Nikt nie zgadł!!!

- Uwaga, uwaga! Sensacja! Nowy model Amigi! Tym razem skok w numeracji jest duży - ma to być Amiga 5000. Komputer ten będzie reklamowany pod hasłem: „Wyjdźmy z ery komputera łupanego.Nowa Amiga ma bić wszystkie maszyny jakie do tej pory ukazały się na rynku. Czekamy!

- Dotarły do nas z Warszawy dwa nowe wirusy: „Butonic wirus" i „The smily cancer" grupy Centurion. Obydwa dogrywają się na dysku w postaci zbioru (file virus). Pierwszy z nich jest nieudaną przeróbką starego „Byte Warrior". Drugi dość skutecznie zatruwa życie posiadaczom komputera. Na jednym z następnych wydań naszego PDP znajdzie się program wykrywający te żyjątka. Butonic'a znaleźliśmy na grze „World Cup 90", natomiast „Uśmiechniętego raka" (The smily cancer) w angielskiej wersji gry „Loom". Sprawdźcie swoje dyski!.

- W jednym z zachodnich pism znaleźliśmy ogłoszenie:
  „Zamienię Atari 520ST na Commodore 128 lub C-64".
  W szczecińskim „Kurierze" pojawiło się natomiast ogłoszenie następującej treści:
  „Zamienię Atari 800XL + drive + dyski na motor. Dopłata do uzgodnienia." Popularność Atari rośnie?!

- Uwaga wszyscy posiadacze Amigi 2000 i karty Bridgeboard! Ze źródeł dobrze poinformowanych dowiedzieliśmy się, że w Krakowie powstaje przystawka umożliwiająca korzystanie ze stacji 5.25" jako DF1:!

- Francja od 17 października 1990 roku posiada nowe prawo skierowane przeciwko piractwu. Zabronione jest nie tylko łamanie i rozpowszechnianie pirackich kopii, alr również ich posiadanie! Te same prawa dotyczA instrukcji do programów. Można więc posiadać tylko jedną - oryginalną wersję programu.
  Nalot policji na dom może się przytrafić w każdej chwili na żądanie kompanii softwarowej czy choćby donosu. Złapanemu piratowi grozi od 3 miesięcy do 2 lat pozbawienia wolności lub 2000 do 40000DM kary. Gdyby u nas były takie prawa Polska w ciągu miesiąca wyszła by z długów!

Hi-Man

JAK MR.F. UNIKNĄŁ PODATKÓW CZYLI FUNDACJA JEST DOBRA NA WSZYSTKO

Na początku był chaos i nieliczne sprowadzane przez prywatnych importerów do Polski Commodore i Amigi. W końcu, o kilka lat za późno, CEL (czyli europejski oddział CBM) postanowił zainteresować się rynkiem w Polsce.

Wyznaczono dealera - Anwara Fancy (USA) i od Targów Poznańskich 1988 rozpoczęła się ekspansja(?) Commodore na Polskę. Początkowo pan F. działał sam. Później postanowił (wykorzystując fakt, że wszelakie fundacje są mile widziane zarówno przez władze jak i przez urzędy skarbowe) założyć Fundację Edukacji Technologicznej. Miała ona wspomagać dzieci niepełnosprawne - jednocześnie zajmując się serwisem Commodore oraz promocją tego komputera na polskim rynku.

Przez prawie 2 lata niezbyt wiele, z rozmaitych powodów, z tego wychodziło. Przełomowym momentem okazał się październik 1990. W miesiącu tym oddzielono dominujący w tym „małżeństwie z rozsądku" serwis od Fundacji. Od tego momentu działalność tej ostatniej zaczęła być zauważalna na rynku.

Na nowej drodze życia Fundacja zdążyła zorganizować dwa COMMODORE SHOW (w Warszawie i Krakowie), oraz wypuścić na rynek nieco, tak potrzebnej każdemu początkującemu użytkownikowi komputera, literatury po polsku.

Zamierzenia Fundacji są dość ambitne. FET pragnie udostępnić posiadaczom Commodore i Amigi zarówno literaturę, jak i oprogramowanie. Nie będą to jednak gry, bowiem Fundacja chcąc działać zgodnie z prawem nie może sprzedawać pirackich, nielicencjonowanych kopii, zaś cena licencjonowanych oryginałów przekraczałaby możliwości finansowe miłośników gier. Ponadto, założeniem Fundacji jest podniesienie kultury informatycznej Commodore'owców, a od psucia joysticka nikt jeszcze ogłady nie nabrał.

Zamiast gier będą więc polskie programy edukacyjne oraz programy użytkowe skierowane w stronę „small businessu", bowiem przedsiębiorcy i rzemieślnicy zaczynają stanowić coraz liczniejszą grupę użytkowników Commodore. Marzeniem zarówno władz Fundacji, jak i osób z nią współpracujących jest model węgierski, gdzie każde przedsiębiorstwo zatrudniające poniżej 100 osób jest w tej chwili obsługiwane przez C64.

Programy w Fundacji są dostępne na dyskietkach 5.25" lub na cartridge'ach. Spośród ciekawszych można wymienić PRACTICALC (baza danych), WARSAW BASIC (znany już nie tylko w Polsce system operacyjny), cartridge HELP (najczęściej potrzebne użytki), PRINTPL (edytor tekstu z polskimi literami - zarówno na ekran jak i na drukarkę; szybki wydruk draft), SŁOWNIK JĘZYKA POLSKIEGO (wspomagający wordprocessor). Są to programy na C64. W niedługim czasie pojawią się również programy na Amigę.

Prócz programów FET ma zamiar wydać dużą ilość literatury. I znowu mamy sytuację podobną do programów. Literatura nie może być słabą kserokopią pirackiego, jakże często nieudolnego tłumaczenia. W tej chwili Fundacja dysponuje autorskimi książkami pozwalającymi na opanowanie w podstawowym zakresie C64 i Amigi oraz BASIC'a C64, zaś bardziej zainteresowanym poleca książkę mówiącą o tym jak rozbudować interpreter C64.

Trzecim nurtem pracy jest wprowadzenie tzw. retailerów, czyli osób, które wspomagają pracę Baltony przy sprzedaży Commodore, jednocześnie służąc konsultacjami osobom, które nabyły Commodore w sklepach Baltony.

Fundacja ma też zająć się prowadzenie kursów tematycznych na różnych poziomach zaawansowania dla użytkowników C64 i Amigi. Już w tej chwili dzwoniąc lub pisząc do Fundacji możesz uzyskać konsultacje w zakresie interesującego Cię problemu z dziedziny C64 lub Amigi.

Fundacja prowadzi także klub użytkowników komputerów Commodore. Każdy członek klubu ma prawo oczekiwać od klubu pomocy w sprawach dotyczących jego komputera. UWAGA: klub nie odpisuje na listy typu:„Przyślijcie mi najnowsze gry na C64".

Tak bogaty program byłby niemożliwy do zrealizowania (w tej chwili w Fundacji pracuje 5 osób) gdyby nie współpraca z osobami, które „w temacie Commodore" mają coś do powiedzenia. Fundacja zapewniła sobie już współpracę kilku takich osób. Ale nadal nie jest to zbyt wiele. Zdajemy sobie sprawę, z tego, że Fundacja nie ma zbyt dobrej reputacji wśród „doświadczonych" Commodore'owców. Jednak czas bajtkowej czapy już się skończył. Pomóżmy Fundacji - kiedyś Fundacja pomoże nam. Dla zainteresowanych (zarówno tych, którzy zechcą wesprzeć Fundację , jak i tych, którzy chcieliby skorzystać z jej usług) podaję kontakt:

FUNDACJA EDUKACJI TECHNOLOGICZNEJ
ul. Okrężna 3
02-916 WARSZAWA
tel. 0-22/424430 wew.18

Tym, którzy nie wierzą ani poczcie, ani telefonom i zechcą się udać do Fundacji osobiście - polecam autobus pospieszny B (przystanek naprzeciw DT Centrum) w kierunku Wilanowa (wysiadać na przystanku Goraszewska), następnie kładką na drugą stronę i 50m na południe.

Miss Shareware

MAPA PAMIĘCI - CZĘŚĆ V

W dzisiejszym artykule przedstawiamy fragment pamięci rozpoczynający się od komórki o adresie 1369, a zakończony komórką o adresie 65535.

Komórki do adresu 2045 włącznie, wykorzystane są jako robocze i zmienne systemu operacyjnego. Organizacja przestrzeni powyżej adresu 2048 zależy od rozmiaru pamięci zainstalowanej w komputerze oraz od trybu (graficzny/tekstowy), w jakim aktualnie pracuje.

W komputerach posiadających 64 kB RAM w obszarze od adresu 32768 do 64767 ($8000-$FCFF) występuje jednocześnie pamięć ROM i RAM. Oznacza to, że w tej przestrzeni jednemu adresowi fizycznie odpowiadają dwie komórki pamięci. Ponieważ procesor może w danej chwili realizować operację odczytu/zapisu tylko jednej komórki, zastosowano dodatkowe układy i podprogramy włączające ten blok pamięci, który jest aktualnie potrzebny. Dla użytkownika proces ten jest niezauważalny i dzięki niemu podczas pracy w BASIC'u ma do dyspozycji 60 kB wolnej pamięci, chociaż interpreter BASIC'a, system operacyjny oraz rejestry urządzeń we/wy zajmują 32kB.

W komputerach Plus4 sytuacja jest jeszcze bardziej skomplikowana, ponieważ posiadają one dodatkowe pamięci ROM zawierające wbudowane oprogramowanie użytkowe.

Tak jak w poprzednich częściach artykułu przejdę teraz do bardziej szczegółowego omówienia wybranych komórek.

W komórkach 1383-1510 przechowywane są łańcuchy przypisane klawiszom funkcyjnym, ich długości zawierają komórki 1375-1382. Łańcuchy te są zapisane po kolei, bez separatorów. Zmieniając za pomocą POKE treść przypisań należy jednocześnie zaktualizować długości.

Komórki 2021-2024 zawierają współrzędne narożników aktualnego okna ekranu.

O sposobie ich wykorzystania pisałem w artykule „C-16 i okna" (64 PLUS 4 nr 2/90).

Zawartość komórki o adresie 2025 decyduje o działaniu scrollingu. Normalnie wynosi ona 0. Wpisanie:

```
10 POKE 2025,128:REM ESC-M
```

wyłącza scrolling. W tym trybie, po wypełnieniu ekranu, kursor jest przenoszony w pozycję HOME i dalsze znaki są wypisywane od tego miejsca. Taki sam efekt wywołuje wprowadzenie sekwencji ESC-M. Normalny scrolling przywrócić można sekwencją ESC-L lub wpisując:

```
20 POKE 2025,0:REM ESC-L
```

Standardowo wypisywane na ekranie znaki kasują te, które były już tam wyświetlone. Wprowadzając sekwencję ESC-A lub wpisując :

```
30 POKE 2026,255:REM ESC-A
```

włączamy tryb, w którym wypisywane znaki powodują rozsuwanie istniejącego tekstu. Normalną pracę przywraca wpisanie:

```
40 POKE 2026,0:REM ESC-C
```

Przy wyświetlaniu długich listingów przewijanie tekstu przez ekran (scrolling) można zatrzymać kombinacją klawiszy CTRL-S. Jeżeli wpiszemy:

```
50 POKE 2039,255
```

to przewijania nie można zatrzymać tym sposobem. Normalnie zawartość tej komórki wynosi 0.

Ostatnią komórką, której funkcję przedstawię, jest komórka o adresie 2040. Jej zawartość decyduje o tym, jaki blok z nakładających się (powyżej adresu 32768) obszarów pamięci ROM/RAM jest dostępny dla monitora TEDMON. Wpisanie POKE 2040,128 powoduje, że funkcje TEDMON czytają pamięć RAM. Czytanie pamięci ROM przywraca wpisanie POKE 2040,0.

Obszar pamięci 64768-65343 ($FDOO-FF3F) zajmują rejestry układów we/wy.

Najważniejszym z nich jest układ TED zajmujący adresy 65280-65311 ($FF00-$FF1F). Rejestry tego układu opiszę w oddzielnym artykule.

ADRES HEX	ADRES DEC	ETYKIETA	OPIS
04F4	1268	TMPTRP	czasowy rejestr dla TRAP
04F5-04F6	1269-1270	ERRTXT	pamięć pośrednia dla TRAP
04F7	1271	OLDSTK	wskaźnik stosu dla ON ERROR GOTO
04F8-04F9	1272-1273	TMPTXT	pamięć wskaźnika tekstu BASIC (D0)
04FA-04FB	1274-1275	TMPLIN	pamięć numeru wiersza
04FC-04FF	1276-1279		obszar roboczy dla SOUND
0500	1280	USRPOK	rozkaz skoku JMP
0501-0502	1281-1282	USRADD	adres USR (low/high)
0503-0507	1283-1287	RNDX	wartość początkowa dla RND
0508	1289	DEJAVV	wskaźnik: start lub restart
0509-0512	1289-1298	LAT	tabela numerów zbiorów logicznych
0513-051c	12991308	FAT	tabela numerów urządzeń
051D-0526	1309-1318	SAT	tabela adresów wtórnych
0527-0530	1319-1328	KEYD	bufor klawiatury (FIFO)

0531-0532	1229-1230	MEMSTR	wskaźnik: adres początku RAM dla systemu operacyjnego
0533-0534	1331-1332	MEMSIZ	wskaźnik: adres końca RAM dla systemu operacyjnego
0535	1333	TIMOUT	wskaźnik: timeout (przerwanie czasowe) z (opcjonalnej) szyny IEC
0536	1334	FILEND	1=osiągnięty koniec zbioru
0537	1335	CTALLY	ilość znaków w buforze (dla read i write)
0538	1336	CBUFVA	ilość w sumie ważnych znaków w buforze (tylko read)
0539	1337	TPTR	wskaźnik następnego znaku w buforze (dla read i write)
053A	1338	FLTYPE	typ zbioru kasetowego
053B	1339	COLOR	aktywny bajt atrybutu
053C	1340	FLASH	wskaźnik: znak miga ($00=nie); nie używany
053E	1342	HIBASE	początek video-RAM (strona)
053F	1343	XMAX	wielkość bufora klawiatury
0540	1344	RPTFLG	wskaźnik: powtarzanie klawiszy ($80=wszystkie, $40=żaden, $00=tylko DEL,CRSR,SPC)
0541	1345	COUNT	szybkość licznia dla powtarzania
0542	1346	DELAY	licznik opóźnienia powtarzania
0543	1347	SHFLAG	wskaźnik: klawisze SHIFT,CTRL,C=
0544	1348	LSTSHF	poprzednia wartość SHFLAG
0545-0546	1349-1350	KEYLOG	wskaźnik tabeli dekodowania klawiatury
0547	1351	MODE	$80=SHIFT nie działa, $00=działa
0548	1352	AUTODN	automatyczne przewijanie, 0=włączone
0549	1353	LINTMP	pamięć pośr. podczas wyj. na ekran
054A	1354	ROLFLG	pamięć pośr. podczas wyj. na ekran
054B	1355	FORMAT	pamięć robocza TEDMON
054C-054E	1356-1358	MSAL	dla asemblera
054F	1359	WRAP	pamięć chwilowa dla asemblera
0550	1360	TMPC	j.w.
0552	1361	DIFF	j.w.
0552	1362	PCH	licznik programu (high)
0553	1363	PCL	licznik progr.(low)
0554	1364	FLGS	rejestr wskaźników procesora
0555	1365	ACC	akumulator procesora
0556	1366	XR	rejestr X procesora
0557	1367	YR	rejestr Y procesora
0558	1368	SP	wskaźnik stosu procesora
0559-055C	1369-1372		komórki robocze TEDMON
055D	1373	KYNDX	licznik znaków łańcucha przypisanego do klawisza funkcyjnego
055E	1374	KEYIDX	wskaźnik łańcucha przypisanego j.w.
055F-0566	1375-1382	KEYBUF	długość łańcuchów przypisanych j.w.
0567-05E6	1383-1510	PKYBUF	miejsce przechowyw. łańcuchów przypisanych j.w.
05E7-05EB	1511-1515		komórki robocze DMA
05EC-06EB	1516-1771	SAVRAM	strona pamięci wykorzystywana przez proc. bankowania
05F0-05F1	1520-1521	LNGJMP	long-jump (adres)
05F2	1522	FETARG	long-jump (akumulator)
05F3	1523	FETXRG	long-jump (rejestr X)
05F4	1524	FETSRC	long-jump (rejestr statusu)
05F5-065D	1525-1629	AREAS	obszar RAM dla proc. bankowania
065E-06EB	1630-1771	APECH	obszar RAM dla syntetyzatora mowy
06EC-07AF	1772-1967	STKTOP	pseudostos dla interpretera BASIC
07B0-07CC	1968-1996		obszar rob. procedur współpr. z magnet.

Wojciech Wasilewski

A s s e m b l e r   6 5 1 0

Lekcja 1

Dotarło do nas wiele listów, w których prosicie o zamieszczenie kursu assemblera 6510. Długo wahaliśmy się przy podejmowaniu decyzji. W Polsce można w tej chwili znaleźć dość dużo publikacji na temat tego procesora. Ich wspólną cechą jest jednak brak odpowiedniej liczby przykładów, które w sposób wyczerpujący wyjaśniałyby podane rozkazy.

Dlatego też w końcu zdecydowaliśmy się rozpocząć cykl artykułów, które będą miały na celu przybliżenie czytelnikom tego procesora, objaśnienie poszczególnych rozkazów i wprowadzenie w tajniki programowania w języku maszynowym. Myślę, że cykl ten w połączeniu z równolegle prowadzonym przez Polonus'a pt. „Jak zrobić własne demo" będzie bardzo pomocny dla ambitniejszych użytkowników Commodore 64, a wielu graczy przekona, że pisanie własnych programów sprawia o wiele więcej przyjemności niż bicie kolejnych rekordów w grach.

Zanim zaczniemy naukę na dobre proponuję Wam abyście zaopatrzyli się w niezbędne programy i kupili jedną dobrą książkę. Te programy to:

> 1. Dowolny monitor języka maszynowego

> 2. Dowolny assembler (najlepiej Macroassembler lub XAŁŁ Assembler)

W przypadku monitora najlepszym wyjściem jest zakup takiego cartridge, który posiada wbudowany na stałe debugger (czyli innymi słowy właśnie monitor). Tu polecam trzy modele obecnie najpopularniejsze na naszym rynku: Final Cartridge II/III i Action Replay.

Zdobycie assemblera może być trudniejsze (handlarzy interesują tylko gry) dlatego też na jednym z dysków „64PLUS 4 FDP" ukaże się wybór najlepszych assemblerów dostępnych na C-64. O tym kiedy to się stanie powiadomimy osobno.

Książka, którą polecam wszystkim bardziej zainteresowanym to „Commodore 64" autorstwa Bohdana Frelka. Wydały ją Wydawnictwa Naukowo Techniczne w 1988 roku. Jest to w zasadzie jedyna poważna i fachowa pozycja na polskim rynku. Autor opisuje w niej system C-64, wyjaśnia zasadę pracy komputera, zahacza o język maszynowy, grafikę i muzykę. Bardziej jednak skoncentrował się na stronie technicznej maszyny i wyjaśnienia niektórych procesów zachodzących w jej wnętrzu są niezrozumiałe dla przeciętnego użytkownika.

Nasz kurs prowadzimy tak aby nawet „zupełnie zielony" użytkownik mógł zacząć pisać własne programy.

Nie potrafię określić ile numerów zajmie mi opisanie całego języka maszynowego. W dużej mierze zależy to od czytelników. Czekam na Wasze listy, pytania i propozycje. Pamiętajcie jednak jedno - tylko uczciwa praca z maszyną da Wam biegłą znajomość assemblera. „Podcinanie" cudzych procedur, zmiany napisów w programach to tylko ośmieszanie swojego nazwiska czy ksywy. „Praca czyni mistrza" i to jest -niestety -prawda.

Języki programowania

Nie będą nas tu interesowały same języki lecz ich podział i to dość specyficzny. Wszystkie bowiem języki programowania można podzielić na: maszynowe (assemblery) i resztę! Tak, to nie jest żart i zaraz to wyjaśnię.

Otóż sercem komputera jest mikroprocesor. Właśnie to on wykonuje wszystkie operacje, zlecone przez użytkownika. Każdy z mikroprocesorów posługuje się swoim własnym językiem, który nazywa się językiem maszynowym lub assemblerem. Jak to się więc dzieje, że nasz C-64 potrafi realizować programy w Basic'u? Przecież nie jest to język, w którym pracuje procesor 6510. Sprawa jest prosta.

Tak naprawdę Basic czy każdy inny język (C,Pascal itp.) to specjalny program, który przekłada instrukcje danego języka na assembler. Jeśli zlecimy naszemu komputerowi np. wykonanie PRINT „64 PLUS 4" to program Basic'a rozłoży tę instrukcję na grupę rozkazów assemblera i wtedy dopiero ją wykona. Już teraz widać, że maszynówka (popularna nazwa języka maszynowego) jest znacznie szybsza od Basic'a czy wszystkich innych języków wyższego poziomu. Co to są języki wyższego poziomu? Otóż są to właśnie te programy przekładające skomplikowane instrukcje na prosty język procesora. Czemu powstały? Aby ułatwić programistom pracę. Przykładowo jedna instrukcja PRINT zajmuje kilkadziesiąt instrukcji w maszynówce. Czemu więc programiści cały czas zwracają się ku językom niższego poziomu? Otóż jedynie programowanie w assemblerze daje pełną kontrolę nad całym komputerem, ogromną szybkość wykonywanych operacji i pozwala na wykorzystanie możliwości maszyny w 100% (czasem nawet w ponad 100%!!!)

Systemy liczbowe

Pora zająć się systemami liczbowymi w jakich potrafi pracować nasz procesor. W zasadzie jest on jeden - system dwójkowy zwany również binarnym. My jednak będziemy posługiwali się również systemem szesnastkowym, który potocznie nazywa się heksadecymalnym.

Jak to wygląda w praktyce? Otóż każda liczba zapisana w pamięci komputera składa się z bitów czyli z podstawowych jednostek informacji. Bit może przyjmować wartości 0 lub 1. Dwie liczby - a więc - system dwójkowy. Działa on na takiej samej zasadzie jak stosowany przez wszystkich system dziesiętny.

Przykładowo liczbę 47 możemy zapisać w ten sposób:

4 7
$4*10^1 +$ $7*10^0$

Jak widać podstawą jest liczba 10. A gdy podstawą będzie liczba 2 - zapis będzie wyglądał następująco:

1 0 1 1 1 1
$1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 47$

A więc liczba %101111 (znak % oznacza, że używamy systemu binarnego) w systemie dziesiętnym (dec) równa się 47.

W podobny sposób można stworzyć dowolny system liczenia np. o podstawie 3, 7 czy 20. Nas jednak będzie interesował jeszcze tylko jeden - szesnastkowy. Skoro w systemie dwójkowym posługujemy się dwoma cyframi (0 i 1), w dziesiętnym dziesięcioma (0-9), to logicznie rzecz biorąc w szesnastkowym tych cyfr będzie szesnaście. Jednak co z brakującymi 6 cyframi (znamy przecież ich tylko dziesięć)? Poradzono z tym sobie zastępując brakujące cyfry kolejnymi literami alfabetu. Oznacza to, że po 9 będzie A, potem B i tak dalej aż do F, które równa się dziesiętnie 15. Teraz już łatwo nam będzie określić ile równa się liczba $F3 (znak $ oznacza liczbę heksadecymalną)

$ F 3
$15*16^1 + 3*16^0 = 243$

i jeszcze jeden przykład:
$1AD=429 bo:

$ 1 A D
$1*16^2 + 10*16^1 + 13*16^0 = 429$

Może Wam wydawać się, że używanie takich systemów jest bez sensu, bo tylko komplikuje życie. Ręczę Wam jednak, że po pewnym czasie będziecie woleli liczyć w szesnastkowym i kląć będziecie system dziesiętny.

Na razie postarajcie się jednak przyswoić sobie wiadomości z tego artykułu, a za miesiąc zaczniemy pisać pierwsze własne procedury.

Sambor Kuźma

Programy ciekawe, zwariowane i takie sobie

Jak zrobić płynne przesuwanie napisów w Basic'u? Zapewne wielu z Was łamało sobie głowę nad tym problemem. Pewnie zrobiliście programik, który przesuwał jedną literę, stwierdziliście, że wygląda to okropnie i daliście sobie spokój.

Ale jednak można!!! I to nie tylko jedną literkę lub napis, ale cały ekran. Tak, tak płynnie sunący np. w prawo i to tylko dzięki krótkiej procedurze w Basic'u.

```
10 FOR A=1024 TO 2023
20 POKE A,0
30 NEXT A
40 FOR A=0 TO 7
50 POKE 53270,A
60 NEXT A
70 GET A$: IF A$=„ " THEN POKE53270,8:END
80 GOTO 40
```

I co? Działa? Oczywiście, że tak. Tylko... Wystarczy z programu usunąć linie 10 do 30 i wtedy okazuje się, że to jest oszustwo. Program daje oczekiwany efekt tylko wtedy, gdy cały ekran (lub cała jedna linia) jest zapełniony tym samym znakiem. Myślę, że jeśli już teraz poeksperymentujecie z komórką 53270 ($D016) to potem będzie Wam łatwiej zrozumieć zasadę działania scroll'a.

W dalszym ciągu czekam na Wasze programy - ciekawe, zwariowane i takie sobie. Nie krępujcie się - ta rubryka jest dla Was otwarta.

Hi-Man

Jeszcze o możliwościach dźwiękowych C-64 !

C-64 to komputer o niezwykle bogatych możliwościach generowania dźwięku. Pisano już o tym wielokrotnie. Niewtajemniczonych spieszę poinformować, że mikrokomputer ten posiada trzy niezależne generatory mogące wytwarzać dźwięki w zakresie 8 oktaw (o kształcie fali: prostokątnej, piłokształtnej lub trójkątnej) oraz szum. Do dyspozycji każdego z nich mamy programowane generatory obwiedni tzw. ADSR. C-64 posiada także zestaw filtrów oraz możliwość regulowania współczynnika wypełnienia dla przebiegu prostokątnego.

Nie każdy wie jednak, że nasz Commodorek potrafi poddawać obróbce sygnał doprowadzony z zewnątrz np. z magnetofonu. Aby móc tego dokonać należy najpierw sporządzić specjalny kabel. Kabel ten musi łączyć wyjście magnetofonu (lub innego źródła sygnału) z otworem nr 5 znajdującym się w gnieździe Audio/Video. Czynność ta powinna być wykonana starannie aby nie spowodować zwarcia wyjścia dźwięku z masą. Mogłoby to się okazać zgubne dla układu SID.

Po wykonaniu wszystkich połączeń należy włożyć do magnetofonu jakąś kasetę z muzyczką, wcisnąć klawisz PLAY i rozpocząć eksperymenty.

W celu usłyszenia wprowadzanego do C-64 dźwięku w głośnikach należy ustawić głośność generatorów na wartość maksymalną za pomocą instrukcji:

```
POKE 54272+24,15
```

Słyszany w głośniku dźwięk można poddawać takiej samej obróbce jak dźwięk pochodzący z generatorów. Najciekawsze, a zarazem najbardziej przydatne wydaje się być filtrowanie. C-64 wyposażony został w cztery następujące filtry:

- dolnoprzepustowy (wytłumia dźwięki o wysokich częstotliwościach)
- pasmowy (przepuszcza dźwięki o częstotliwościach zawartych w pewnym paśmie),
- górnoprzepustowy (wytłumia dźwięki o niskich częstotliwościach),
- rezonansowy (zwiększa głośność dźwięków o zadanej częstotliwości).

Każdy z w/w filtrów wymaga podania pewnej charakterystycznej dla niego częstotliwości. Młodszy bajt częstotliwości wstawiamy do komórki o adresie 54272+21 natomiast starszy do komórki o adresie 54272+22. Przykładowo :

```
POKE 54272+21,0
POKE 54272+22,100
```

Za załączenie filtrów odpowiedzialny jest bit 3 w komórce o adresie 54272+23 np:

```
POKE 54272+23,8
```

Filtr dolnoprzepustowy załączamy za pomocą bitu 4 komórki o adresie 54272+24 np:

```
POKE 54272+24,1*16 +15
```

Filtr pasmowy załączamy za pomocą bitu 5 komórki o adresie 54272+24 np:

```
POKE 54272+24,2*16+15
```

Filtr górnoprzepustowy załączamy za pomocą bitu 6 komórki o adresie 54272+24 np:

```
POKE 54272+24,4*16+15
```

Skuteczność działania filtru rezonansowego ustalamy za pomocą bitów 4-7 komórki o adresie 54272+23 np:

```
POKE 54272+23,15*16+8
```

Oprócz filtrowania sygnał może być poddany procesowi modulacji amplitudowej a także łączony z dźwiękiem generatorów.

Wszystkie te i inne możliwości stawiają C-64 dość wysoko wśród 8-bitowych komputerów domowych dając pole do popisu użytkownikom oraz dostarczając wiele zabawy.

JAMESOFT

Zapraszamy wszystkich do udziału w stałym konkursie pod hasłem:

Najlepszy program miesiąca

W konkursie udział mogą brać wszyscy, którzy nadeślą własne, nigdzie nie publikowane prace.

Tematyka programów dowolna.

Konkurs rozgrywany jest osobno dla komputerów C-16 i C-64.

Teksty programów należy nadsyłać na adres redakcji na dyskietce lub w postaci czytelnego rękopisu (dyskietki będą przez redakcję zwracane).

Objętość programu wraz z opisem i komentarzem nie powinna przekraczać 4 stron maszynopisu.

Raz w miesiącu Sąd Konkursowy wybierze najlepsze programy przyznając ich autorom dwie główne nagrody po **500.000** zł każda. Decyzje Sądu Konkursowego są nieodwołalne. Oprócz zdobycia głównej nagrody autorzy mają szansę na publikację swych prac na łamach naszego pisma.

Pracę prosimy podpisywać imieniem i nazwiskiem oraz dokładnym adresem autora.

Redakcja

JAK ZROBIĆ WŁASNE DEMO (1)

Na życzenie Czytelników rozpoczynamy cykl artykułów, w których przybliżymy zasady pisania własnych programów demonstracyjnych, zwanych popularnie „demami". Tych, którzy nie są zainteresowani w napisaniu dema, zawarte tu informacje i przykłady procedur na pewno pomogą w lepszym wykorzystaniu komputera.

Przykłady procedur zostały napisane przy użyciu MacroAssemblera w taki sposób, by nie sprawiały kłopotów przy przenoszeniu ich pod inny, posiadany przez Czytelnika assembler.

Ze względu na dość duży stopień skomplikowania niektórych procedur wskazane jest przynajmniej niewielkie własne doświadczenie w pisaniu programów w kodzie maszynowym dla C-64. Początkujących odsyłam na stronę 8 naszego pisma do artykułu pod tytułem: „Assembler 6510".

Pierwszym tematem, którym się zajmiemy będzie zagadnienie wykorzystania systemu przerwań komputera przy tworzeniu własnych programów. Czym są przerwania? Na pewno wiecie, że Wasz Commodore jest w stanie wykonywać tylko jedną czynność w tej samej chwili, prawda? Może zróbmy mały eksperyment i napiszmy w BASIC'u:

```
PRINT TI$
```

i wciśnijmy RETURN. Posłuszny komputer poda nam czas w godzinach, minutach i sekundach, który upłynął od momentu włączenia komputera (dla niezorientowanych: przykładowa odpowiedź komputera „001245" oznacza 00 godzin, 12 minut i 45 sekund). fakt zaliczania mijającego czasu wskazuje na to, że nasz C-64 coś jednak sobie „na boku" robi, niezależnie od naszej, np. edycji programu w BASIC'u. Jak to jest możliwe? Po prostu na dany sygnał procesor „odrywa się" od swojego głównego zajęcia (np. obsługa edytora BASIC'a) i wykonuje szereg stosunkowo prostych, ale ważnych z punktu widzenia pracy systemu procedur, jak m.in. odczyt klawiatury i obsługa wspomnianego wcześniej zegara. Po wykonaniu tych zadań procesor powraca do wykonywania przerwanej pracy głównej.

Zawarta w pamięci ROM procedura obsługi przerwań pozwala bardziej zaawansowanemu użytkownikowi na wykorzystanie istniejącego systemu przerwań do własnych zastosowań. W tym celu projektanci systemu operacyjnego wykorzystali technikę zapisu procedury IRQ (przerwań) w postaci dwubajtowego wektora, który znajduje się w pamięci RAM, gdzie może być wygodnie modyfikowany do własnych potrzeb. Wektor ten stanowią dwie kolejne komórki o adresach $0314 i $0315 (młodszy i starszy bajt adresu IRQ). Standardowo wskazuje on adres $EA31. W uproszczeniu wygląda to tak: procesor otrzymuje sygnał o konieczności wykonania przerwania, przerywa swoją dotychczasową pracę, zapamiętuje stan licznika rozkazów, flag i poszczególnych rejestrów a następnie wykonuje rozkaz JMP ($0314), który standardowo kieruje go pod adres $EA31, gdzie następuje wykonanie wyznaczonych operacji a na koniec pobranie zapamiętanych danych (flagi, licznik i rejestry) i powrót do miejsca w którym „zaskoczyło" go przerwanie. Proste? „Podwieszenie się" pod system polega więc na skierowaniu wektora IRQ na naszą własną procedurę obsługi przerwań. W schemacie assemblerowym wyglądać może to np. tak:

```
ORG $5000          ;nie wiem czemu lubię ten adres...
           SEI
           LDA  #<IRQ
           STA  $0314
           LDA  #>IRQ
           STA  $0315
           CLI
           RTS
:IRQ       JSR  naszaprocedura1
           JSR  naszaprocedura2
           ....
           ....
           JSR  naszaproceduraileśtam
           JMP $EA31    ;skok do ROM
```

Prawda że proste? Musimy zadbać tylko to, by czas potrzebny na wykonanie naszych procedur nie był zbyt długi, ponieważ wtedy czas przeznaczony na wykonywanie programu głównego będzie proporcjonalnie skracany (aż do zera?). Procedurę przerwań niekoniecznie trzeba kończyć skokiem pod adres $EA31, ale wtedy trzeba samemu zadbać o prawidłowe zakończenie IRQ (wzór w ROM pod adresem $EA7E).

Jest to tylko jeden z trybów przerwań w Commodore 64, które można wykorzystać do własnych celów. Innym, i chyba najczęściej wykorzystywanym jest tryb przerwań rastrowych (w/g. innych: przerwań graficznych VIC'a, jak zwał, tak zwał-chodzi o to samo).

Na czym polega specyfika przerwań rastra? Od poprzednich różnią się w zasadzie tylko źródłem sygnałów wywołujących przerwanie. O ile w pierwszym przypadku źródłem był wbudowany generator kwarcowy,to w drugim do przerwań pobudza sam VIC (Video Interface Chip - układ graficzny w C-64). Jakie są z tego korzyści? najważniejsza to ta, że można wywołać przerwanie w interesującej nas lini ekranu podczas operacji generowania obrazu telewizyjnego. Dzięki temu przy odrobinie wprawy możliwe stają się płynne animacje (tzn. bez żadnych „schodów", jak np. na IBM'ach czy Amstradach).

Ale jak je wywołać w żadnej lini? Do poinformowania VIC'a, w której lini ma nastąpić przerwanie służą dwie znajdujące się w nim komórki: $D012 i $D011. Przy opisie numeru jednej z 320 linii w systemie PAL (lub 262 w NTSC) jeden bajt nie wystarczy, więc jako dziewiątego bitu, po ośmiu bitach $D012, używa się najstarszego bitu w komórce $D011 (tego o „wartości" $80).

A oto schemat assemblerowy przerwań rastrowych ustawionych np. na linię $32 (początek ekranu tekstowego):

```
ORG   $5000
        SEI
        LDA #$7F         ;wyłącz zegar
        STA $DC0D        ;jako źr. IRQ
        LDA $D011        ;skasowanie
        AND #$7F         ;nieużywanego
        STA $D011        ;9-tego bitu
        LDA #$32         ;wstawienie
        STA $D012        ;numeru lini
        LDA #$01
        STA $D01A        ;włączenie IRQ-VIC
        LDA #>IRQ
        STA $0314
        LDA eIRQ
        STA $0315
        CLI
        RTS

:IRQ    JSR naszaprocedura
        INC $D019        ;potw.IRQ-VIC
        JMP $EA31        ;skok do ROM'u
```

O tym, jakie daje to efekty, możemy się przekonać pisząc podprogram o nazwie „naszaprocedura":

```
:naszaprocedura
        LDX #$40
:loop   LDA $D012
        TAY
        AND #$07
        EOR #$1C
:test   CPY $D012
        BEQ test
        STA $D011
        DEX
        BNE loop
        RTS
```

O tym, jak działa ten efekt napiszę następnym razem omawiając efekty związane z wykorzystaniem przerwań rastrowych przy tworzeniu dem.

Owocnych eksperymentów życzy

Paweł Sołtysiński

Graj aby wygrać

1. RANARAMA

```
POKE  37104,96
      33969,234     energia
      33970,234
SYS   32768
```

2. FIRE TRACK

```
POKE  12285,234
      12286,234     życie
      12287,234
SYS   9216
```

3. EQUALIZER

nacisnąć „C"+RESTORE - przejście do następnej planszy.

4 HE MAN

```
POKE  12651,234
      12652,234     życie
      12653,234
SYS   17610
```

5. HIGHWAY ENCOUNTER

```
POKE  6690,234
      6691,234      pomaga zaktywować
      6692,234      lasertraia
SYS   4103
```

6. DANDY

```
POKE  8764,173
      5697,173      energia
      11911,173
      5718,173
      9800,0
SYS   4371
```

7. ZYRON'S ESCAPE

```
POKE  51591,234
      51592,234     życie
      51593,234
SYS   4166
```

8. 1942

```
POKE  52212,128     życie
SYS   51945
```

9. POOYAN

```
POKE  20672,0
      20673,0       życie
      20674,0
SYS   27528
```

10. BOPN'RUMBLE

```
POKE  2452,165      życie
      4261,x+48     x - poziom startu
SYS   2204
```

Hi-Man

RESET na każdą okazję

Użycie tradycyjnie wykonanego przycisku RESET nie zawsze okazuje się skuteczne - szczególnie jeśli w programie znajduje się tzw. „start modułowy".

Jak funkcjonuje „start modułowy"?

Co to wogóle jest „start modułowy"? By to wyjaśnić, trzeba z bliska przyjrzeć się systemowi operacyjnemu C-64. Po naciśnięciu klawisza RESET, lub po wykonaniu rozkazu SYS 64738 komputer bada zawartość komórek $8004 do $8008. Gdy znajduje się tam napis „cbm80" w formie ciągu znaków COMMODORE-ASCII, następuje skok do procedury o adresie znajdującym się w komórkach $8000 i $8001. Przy czym komórka $8000 zawiera młodszą a $8001 starszą część adresu skoku. W komórkach $8002 i $8003 znajduje się wektor tzw. ciepłego restartu wywoływany naciśnięciem klawisza RESTORE. Jeśli w komórkach $8000, $8001 znajduje się adres początku programu, to po naciśnięciu klawisza RESET nastąpi ponowne uruchomienie programu.

Nazwa „start modułowy" pochodzi stąd, że każdy moduł-cartridge, który powinien wystartować po włączeniu komputera, zawiera w komórkach $8004 do $8008 znaki:„cbm80", a w komórkach $8000 i$8001 adres startu zawartego w nim programu. Użycie przycisku RESET będzie, w takim przypadku,powodować wywołanie programu cartridge'a, a nie planszy tytułowej komputera.

Wyrywamy się.

Ponieważ cartridge zmieniają konfigurację (pobraną z ROM'u) C-64, staje się jasne, że przycisk RESET nie jest w stanie go wyłączyć. Jeśli sygnał EXROM (zewnętrzny ROM) jest przyłączony do masy to C-64 odłącza pamięć RAM od adresu $8000 do $A000 i wstawia w to miejsce ROM modułu (EPROM). Fakt ten wykorzystamy by „wyrwać się " z modułu startowego programów. Jeśli podczas RESET'u sygnał EXROM będzie zwarty z masą, to C-64 będzie szukał w nieistniejącym module znanego nam textu „cbm80". Nie znajdzie go - co spowoduje, że komputer wykona normalną procedurę restartu z skokiem do inicjalizacji BASIC'a (komunikat po włączeniu komputera), dzięki czemu komputer powróci do pierwotnej konfiguracji.

Przełącznik RESET w praktyce.

Jak już wspominaliśmy, podczas resetowania linia EXROM musi zostać połączona z masą (GND). Najłatwiej dokonać tego stosując przełącznik, który zwiera jednocześnie dwie niezależne linie (podwójny przełącznik). Ponieważ sygnał EXROM musi mieć niski potencjał w czasie trwania RESET'u to - przy pomocy kondensatora - tak ograniczymy czas trwania sygnału RESET aby był on krótszy od czasu trwania EXROM. Użyjemy kondensator elektrolityczny (16V/47mikroF) włączając go w obwód przełącznika RESET tj. pomiędzy linię RESET (oznaczona na złączu jako C) oraz jednym ze styków przełącznika. Biegun minus kondensatora musi być po stronie masy.

Funkcjonowanie tego układu jest stosunkowo łatwe do wyjaśninia: po naciśnięciu przełącznika obie okładki kondensatora mają potencjał masy. Reset zostaje uaktywniony. Po krótkim czasie kondensator ulega naładowaniu i linia RESET znajduje się ponownie w stanie wysokim, podczas gdy linia EXROM (nr 9 na złączu) ma potencjał niski aż do zwolnienia przycisku. Aby uzyskać ponowne rozładowanie kondensatora po zwolnieniu przycisku, jest on bocznikowany rezystorem 10kOhm o mocy 1/8Watt.

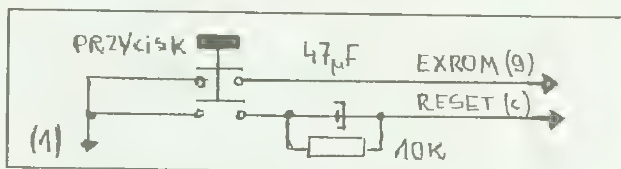
Uwaga - zbytnie ograniczenie czasu trwania impulsu RESET może powodować, że współpracujące stacje dysków nie będą „zauważały" tego sygnału i nie wykonają RESET'u.

Wbudować do wewnątrz czy też wykonać moduł zewnętrzny?

Osobom biegłym w lutowaniu polecamy wbudowanie układu do wewnątrz komputera (uwaga na gwarancję). Schemat układu znajduje się na rysunku poniżej.

Innym możliwym rozwiązaniem jest dołączenie układu poprzez gniazdo rozszerzające. Jednakże wtyk do tego gniazda trzeba wykonać samemu. Do wykonania wtyku potrzebna jest dwustronnie laminowana płytka o rozmiarach 58x50mm. Przy wyłączonym komputerze należy kilkakrotnie wsunąć i wyciągnąć płytkę z gniazda by można było na niej zaobserwować ślady ścieżek. Następnie należy usunąć około 1mm miedzi pomiędzy śladami ścieżek na całej długości złącza po obu stronach płytki. Można to zrobić przy pomocy rylca, punktaka lub ostrego gwoździa oraz linijki. Do przygotowanych ścieżek wystarczy dolutować zgodnie ze schematem pozostałe części i układ jest gotowy.

Zwaracamy uwagę, aby wszystkie czynności przeprowadzić bardzo starannie, wszelkie niedokładności mogą być przyczyną kosztownej awarii komputera.



Schemat układu RESET

Na podstawie 64'er nr 6/85 opr. T.J W.S.

Sampling - coś o nieskończoności dźwięku

Obracając się w świecie komputerowców, muzyków, czy też posiadaczy CD Player'ów można zetknąć się z takim pojęciem jak SAMPLING. Nazwa ta brzmi dla nie wtajemniczonych dość zagadkowo...

Sampling jest to, krótko mówiąc, przetwarzanie dźwięku (lub jakiegokolwiek innego sygnału analogowego) na postać cyfrową. Taka definicja z reguły mówi niewiele, a więc przejdźmy do konkretów. To co nazywamy dźwiękiem jest w rzeczywistości niczym innym jak szybko zmieniającym się w czasie ciśnieniem powietrza rejestrowanym przez nasze ucho. Za pomocą odpowiedniego przetwornika (np. mikrofonu) można te zmiany ciśnienia przetworzyć na odpowiadające im zmiany napięcia elektrycznego. Następnym etapem całej operacji SAMPLING'u jest zamiana napięcia elektrycznego na ciąg liczb. Można przyjąć, że w znikomo krótkim czasie napięcie to nie zdąży się znacznie zmienić. Dla dźwięku słyszalnego przez ludzkie ucho czas taki może wynosić ok. 5 mikrosekund. Nie pozostaje nam zatem nic innego, jak mierzyć co 5 mikrosekund napięcie, a wyniki pomiarów (będące liczbami) zapisywać w kolejnych komórkach pamięci i... proces SAMPLING'u mamy ukończony.

W rzeczywistości wszystkie pomiary dokonywane są przez specjalny układ scalony (tzw. przetwornik analogowo-cyfrowy), a wyniki zapisywane w pamięci komputera. Przebiegiem wszystkich tych operacji steruje specjalny program. Proces przetwarzania dźwięku z postaci cyfrowej na analogową przebiega dokładnie odwrotnie tzn. liczby pobierane z pamięci komputera zamieniane są na odpowiadające im napięcia elektryczne (proces ten jest realizowany przez przetwornik cyfrowo-analogowy), które po wzmocnieniu we wzmacniaczu i podaniu na kolumny głośnikowe dają poprzednio SAMPLE'owany dźwięk.

Dźwięk przechowywany w postaci cyfrowej nosi nazwę SAMPLE, a wartość pojedynczego pomiaru napięcia nosi nazwę próbki.

Powstaje pytanie po co tyle zachodu i co to daje w praktyce ?!

Dźwięk przetworzony w postać cyfrową można w bardzo szerokim zakresie poddawać obróbce bez konieczności stosowania skomplikowanych i bardzo drogich urządzeń. Można tego dokonać posługując się tylko komputerem i odpowiednim programem. W przypadku Amigi takim programem jest Audiomaster. Umożliwia on miksowanie, filtrowanie, tworzenie dowolnego echa, przesłuchiwanie dowolnego fragmentu SAMPLE'a oraz uzyskiwanie gamy wielu innych efektów. Nowoczesny sprzęt muzyczny opiera się w znacznej mierze na cyfrowej obróbce dźwięku.

Trzeba przyznać, że twórcy Amigi poszli za ciosem i wyposażyli ją w cztery przetworniki cyfrowo-analogowe (zwane popularnie czterema generatorami). Tłumaczy to fakt posiadania przez ten komputer niesłychanych walorów muzycznych. Wyspecjalizowane układy pozwalają na odtwarzanie SAMPLE'i bez potrzeby angażowania mikroprocesora w tą pracochłonną czynność. Odtwarzanie SAMPLE'i z różną prędkością umożliwia zagranie dowolnego utworu muzycznego nawet za pomocą odgłosu tłuczonej szyby. Kanały dźwiękowe można łączyć w ten sposób, aby jeden z nich modulował inny itd.

Producenci Amigi nie wyposażyli jej niestety w przetwornik analogowo-cyfrowy, można go natomiast dokupić osobno. Każda próbka jest 8-bitowa (liczby z przedziału od -128 do 127) co daje jakość dźwięku odbiegającą od jakości CD-Player'ów, ale dla przeciętnego ucha nie jest to aż tak istotne.

Programiści i muzycy pracujący na nieśmiertelnym C-64 zawsze marzyli o tym, aby posiadał on czwarty generator, a jeszcze lepiej gdyby mógł odtwarzać SAMPLE. Nie było to jednak takie proste gdyż komputer ten nie był wyposażony w przetworniki cyfrowo-analogowe. Przyszedł jednak czas, kiedy pewien bystry programista wpadł na pomysł jak taki układ zastąpić. Zauważył on, że: jeśli będziemy „bombardować" komórkę pamięci odpowiedzialną za głośność generatorów 4-bitowymi wartościami to otrzymujemy efekt odtwarzania SAMPLE'i. W ten sposób mamy w C-64 czwarty generator. Daje to możliwość używania dźwięków perkusyjnych w muzyce, a także tworzenia wielu ciekawych efektów dla innych celów.

Przykładami programów używających SAMPLE'i są:

* Rockmonitor
* Drummaker
* Pro Drum
* Funky Drummer

oraz wiele innych.

Przetwarzanie sygnałów na postać cyfrową może mieć mnóstwo innych praktycznych zastosowań. Informacja cyfrowa jest odporna na zniekształcenia i przekłamania - ma to ogromne znaczenie we współczesnych systemach łączności. Możliwość zapamiętania sygnałów bardzo ułatwia proces ich analizy.

Graficzne odwzorowanie różnych przebiegów ma duże znaczenie dydaktyczne. Pozwala pokazać zależność barwy dźwięku od rodzaju drgań. Na ekranie otrzymujemy wykres wychylenia w zależności od czasu. Wykres ten jest stabilny w odróżnieniu od obrazu otrzymywanego na oscyloskopie. Pamięć komputera pozwala na „zatrzymanie czasu w miejscu" przez co możemy bardzo dokładnie obserwować szybko zmieniające się przebiegi.

Jak widać z przytoczonych przykładów SAMPLING odgrywa olbrzymią rolę w świecie dźwięków i komputerów.

JAMESOFT

Słownik slangu cz. 1

Po teoretycznych dywagacjach na temat slangu komputerowców pora, tak jak obiecywałem, zająć się wyjaśnianiem poszczególnych wyrazów i zwrotów. Cały nasz słowniczek można by podzielić na dwie części. Pierwsza była by tłumaczeniami angielskich haseł, druga składałaby się z typowo slangowych zwrotów wyłącznie w języku polskim. My jednak pomieszamy te części...
A więc LET'S GO!

Acorn - można śmiało powiedzieć, że to BYŁY komputery, ponieważ obecnie nikt z nich nie korzysta.

Action - jest to pewien rodzaj gier przy których najszybciej padają joystick'i; pewien geniusz przetłumaczył to jako „gry akcji".

Action! - wykrzyknik oznacza, że jest to świetny język programowania dla komputerów Atari, innymi słowy jest to coś bardzo słabego (patrz - Atari).

Add - takim to śmiesznym słówkiem Anglicy oznaczają dodawanie. Jeśli ktoś Ci powiedział „addnij pamiątkę" to znaczy, że masz dodać pamiątkę...(patrz hasło - pamiątka).

Adres - mikroprocesor jest tak mało rozgarnięty, że sam nie wie gdzie ma umieszczać dane. Z tego powodu należy mu podać adres pod który ma je wysłać (kod pocztowy zbędny). Zwrot „ta gierka startuje od dolar tysiąca" nie oznacza wcale ceny wywoławczej tylko właśnie adres startu w pamięci programu (patrz - dolar).

Adventure - rodzaj gier, które w zależności od tego kto je pisał są do przejścia lub nie. Przykładem gry nie do przejścia może być „Operation Stealth", której rozwiązanie czytaliście w jednym z poprzednich numerów.

Agnus - pod tą piękną nazwą kryje się bardzo drogi, specjalizowany układ do Amigi. Niektórzy powiadają, że jest on do wizji, a ja powiadam, że jest do niczego bo mój się niedawno spalił... (patrz - Amiga).

Akumulator - „średniomądra" nazwa rejestru procesora, w którym są wykonywane wszystkie operacje. Zaletą tego rejestru jest to, że nigdy się nie wyładowuje.

Allokacja - proces polegający na zagarnięciu pewnego obszaru pamięci dla własnych celów (task'ów). Na wszelkiego rodzaju zlotach popularnym procederem jest allokowanie dyskietek kolegów. Uwaga ! proces dellokacji może być przykry w skutkach (patrz - Medytujący Guru).

Amiga - najlepszy, najwspanialszy, najładniejszy, najszybszy, najcudowniejszy, najfajniejszy, naj..., naj... komputer na świecie.

And - i.

Atari- komputer stojący w hierarchii tylko trochę niżej niż mój kieszonkowy Casio (patrz - złom).

Available - jeśli jakaś giera jest not available to znaczy, że nie będziesz mógł sobie w nią pograć.

Suplement

Pamiątka - termin oznaczający pamięć komputera. Często używany w zwrotach „Addnij pamiątkę" (czyli użyj programu AddMem) i „mam mało pamiątki" (czyli brak mi rozszerzenia pamięci).

Dolar ($) - na początek: właściciel każdego C-64 posiada na własność 65536 dolarów;
na środek: i tak nic za nie nie kupi;
na koniec: tak naprawdę to chodzi o oznaczenie systemu szesnastkowego.

Rejestr - takie miejsce w procesorze, w którym programista może przechować swoje dane.

Medytujący Guru - najgorsza rzecz jaka może się przytrafić programiście - oznacza ona bowiem utratę kontroli nad systemem.

Złom - popularne określenie komputerów bezwartościowych lub drogich, a beznadziejnych (patrz - Atari).

Hi-Man

PUBLIC DOMAIN PACK C-64

Styczeń
Strona A
- Mega demo grupy „VISION" - MIST2

Strona B
- Preview do gry „UN SQUADRON"
- Preview do gry „PUZZLENOID"
- Preview do gry „TURRICAN"

Luty
Strona A
- TUNE OF MONTH
- LOGO WRITER V 2.0
- FAST CRUELCRUNCH
- WRATH+ (DEMO) [02]
- DREPTACZ BASIC

Strona B
- SWISS CHEESE/CFA
- DISK FAST LOADER

Marzec
Strona A
- FONT GRUB 1.0
- PROJEKTANT DUSZKÓW
- STRZAŁKA 64+
- PIRATEK - GRA
- V4.0 - SYMPHONIES
- CRUISER
- THE FIRST
- COMMERCIAL BREAK
- RELOKATOR 64
- KOREKTOR 64
- FLASH

Strona B
- HOT SHOT nr. 9 (zachodni magazyn fanów)
- BAD NEWS nr. 2 - j.w.
- DEMO - rekord -290 sprite'ów!
- DEMO: NEW INTRO
- DEMO: LET'S DYCP
- KONTAKT CORNER - adresy, kontakty
- NEW FAST - działa z 1541 I 1541 II
- CSLINKER V2.0

O zasadach nabywania naszych Pack'ów czytaj na str. 17.

SPACE ACE

Space Ace to chyba jedna z najbardziej widowiskowych gier na Amidze. Gracz ma wrażenie uczestniczenia w szaleńczym filmie animowanym, w którym właśnie gra główną rolę. Dźwięk, grafika, animacja - wszystko to jest niesamowicie sprawnie zrobione. Gra ma jednak jedną poważną wadę, której nie udało się uniknąć - jest bardzo mało (a w zasadzie wcale) urozmaicona. Wszystkie sekwencje animacji są na stałe wprowadzone do komputera i jedyne co jest wymagane od gracza to poruszenie joystick'iem w ściśle ustalonych momentach.

Gra jest podzielona na 33 sceny, w których ingerencja gracza ograniczona jest najczęściej do jednego ruchu joystick'iem! Jeśli się go nie wykona to zostanie wyświetlona po prostu sekwencja „śmierci" i daną scenę trzeba powtórzyć jeszcze raz. Mimo tych wad warto choć raz zobaczyć całą grę. Zakończenie jest niesamowite!

Poniżej podane są ruchy, które należy wykonać w każdej scenie. Każdy z nich należy zrobić w odpowiednim momencie, dlatego nie przejmujcie się, jeżeli coś Wam nie wyjdzie za pierwszym razem.

Literki oznaczają:
P - prawo
L - lewo
G - góra
D - dół
F - fire w joystick'u.

A więc zaczynamy!

1. P,L,D18. P
2. P,L,L19. G
3. D,G20. F
4. G21. F
5. F22. F
6. P,G23. P
7. P,P24. F
8. D,P25. G,D
9. D,L26. P,D
10. F27. L
11. G28. D
12. G29. P
13. P30. P
14. G31. L
15. P32. P
16. L33. L,P
17. L

Dla leniwych mam również niespodziankę. Po wpisaniu tajnego hasła „DUDEMODEXTER" gra „przejdzie się" sama! Wystarczy usiąść i zmieniając dyski obejrzeć wszystko do końca.

Powodzenia!

Hi-Man

D-Mon Professional

Już od kilku tygodni jestem szczęśliwym posiadaczem monitora języka maszynowego „D-Mon Professional v 1.6", autorstwa Marcina Dudara. Dziś chciałem się z Wami podzielić moimi uwagami o tym programie.

Program został napisany przez człowieka, który na codzień używa monitora do pracy. Wbrew pozorom jest to bardzo ważne. Już od dawna wiadomo, że najlepsze zabezpieczenia piszą hackerzy, a najlepsze scenariusze do gier - gracze. Marcin jest programistą i stąd jego monitor jest naprawdę dobry.

Tym co go wyróżnia spośród wielu innych monitorów to pełno-ekranowy edytor. Ci z Was, którzy pracowali na C-64 i przerzucili się na Amigę wiedzą co stracili. Ekranowy edytor C-64 był bardzo wygodny, a niestety żaden monitor na Amigę nie oferował podobnych możliwości. Teraz jednak mamy 80-cio znakowy edytor, który - wierzcie mi - bardzo ułatwia pracę. Drugą, bardzo istotną różnicą w stosunku do innych monitorów jest to, że D-Mon pracuje na własnym ekranie w multitasting'u! Marcin wyszedł z założenia, że skoro Amiga ma wielozadaniowość to czemu jej nie używać! Odpadają więc problemy z uruchamianiem kilku programów, a opcja OPEN NEW CLI stała się zbędna.

Oprócz klawiszy koniecznych przy edycji ekranowej typu CLR (F10 - kasuje ekran), CLR below cursor (F9 - kasuje ekran pod kursorem) i HOME (F1 - kursor wraca w lewy, górny róg ekranu) Marcin zainstalował kilka nowych funkcji i znacznie rozbudował stare. Pierwszą z nowych opcji jest SET COPPER, która ustawia adres programu dla Copper'a w żądane miejsce pamięci. Klawiszem F7 można ustawić Copper'a dla CLI. Opcja VIEW ASCII wyświetla same znaki ASCII co jest bardzo przydatne przy edycji tekstu np. w demie. Oczywiście w każdej chwili jest możliwa zmiana zawartości pamięci - po to przecież jest monitor. Wystarczy „wjechać" kursorem na żądane miejsce, wpisać nowe wartości i potwierdzić wszystko klawiszem „ENTER". Taka edycja jest o wiele wygodniejsza i szybsza niż edycja liniowa (Timo Rosii monitor). Kolejną nowością jest wbudowanie na stałe do monitora programu Mem Viewer, narzędzia pozwalającego na błyskawiczne przeglądnięcie pamięci w postaci jednego bitplane'u.

Większość starych opcji jest znacznie poprawiona i rozbudowana. Rozkaz FILL ma możliwość wypełniania pamięci całymi grupami bajtów (pattern fill). Opcja LOAD umożliwia załadowanie tylko części pliku! Jeśli chcemy np. wczytać 100 bajtów z pliku zajmującego 100 kB to wystarczy napisać: „nazwa",adres startu 100. Również lepiej jest rozwiązany (w stosunku do monitora T.R.) SAVE. Zamiast podawać długość zbioru do nagrania podaje się adres końca pliku. Jest to o wiele wygodniejsze. Dość ciekawie autor rozbudował funkcję HUNT czyli poszukiwanie w pamięci bajtów. Jeśli chcemy poszukać grupy bajtów np. $FC,$AD,$10, a nie jesteśmy pewni czy w środku na pewno znajduje się wartość $AD to wystarczy zapisać H adres końca $FC ? $10. Znak „?" zastępuje więc wszystkie bajty, które mogą się znaleźć na jego miejscu. Zupełnie nową funkcją jest poszukiwanie tzw. adresów relokowalnych, czyli takich, które są obliczane względem PC (Program Counter). Monitor jest również wyposażony w bardzo dobry kalkulator autorstwa Krzysztofa Kobusa.

To były zalety programu.

D-Mon ma jednak kilka wad. Najpoważniejszą z nich jest brak funkcji REDIRECT OUTPUT. Jest to opcja wręcz konieczna przy poważniejszym zastosowaniu monitora. Jeśli Marcin będzie chciał sprzedać swój monitor jako w pełni profesjonalny - musi koniecznie dołożyć tę funkcję.

W programie udało mi się wykryć kilka tzw bug'ów czyli błędów, a raz nawet monitor zawiesił się!

Testowana przeze mnie wersja D-Mon'a ma symbole v 1.6 i nie jest przeznaczona do rozpowszechniania. Marcin wychodzi ze słusznego założenia, że tylko w 100% sprawdzony i przetestowany produkt ma sznsę na zdobycie popularności . Dlatego też pracuje obecnie nad wersją 2.0, która - miejmy nadzieję - będzie już pozbawiona wad.

Oprócz wielu kosmetycznych poprawek, jak komunikaty o błędach czy przekazywanie adresu po LOAD do monitora, Marcin chce zainstalować szereg nowych opcji i rozbudować stare. Przede wszystkim Mem Viewer otrzyma możliwość zmiany rozdzielczości, modulo, szerokości ekranu, właściciele BIG Agnus'a będą mogli oglądać całe 1MB pamięci. Również za pomocą Mem Viewer'a będzie można kasować i kopiować bloki danych i odtwarzać sample. Mają zostać dobudowane opcje: MEM INFO, AVAIL MEM wraz z instalacją Mutant Led'a 2.0 (protektor przeciw wirusom).

Pozostaje mieć nadzieję, że nowy monitor jak najszybciej przekazany zostanie do dystrybucji.

Sambor Kuźma

PUBLIC DOMAIN PACK AMIGA

Styczeń
- **Programy kompresorów danych**
- **Grafiki Borysa Vallejo**
- **Prezentacja najlepszych muzyczek**
- **INTUITRACKER**

Luty
- **Request player;Multi ripper**
- **3-rd day; Phantasmagoria - demo**
- **Master Seka; Virus Ekspert v1.6**
- **AMOS- programy; Moduły: Killing game show, Upon Me, Let's swing it.**

Marzec
- **Najnowszy i najlepszy program muzyczny PROTRACKER V1.0 (pakiet programowy)**
- **Najlepsze muzyczki: NOW WAT? - DR. AWESOME**
- **AMOS - procedury**
- **DEMO grupy REBELS „TOTAL TRIPLE TROUBLE"**

Zestawy „64 plus 4 PUBLIC DOMAIN PACK" można zamawiać wpłacając na konto: Bank PKO SA Oddział w Bydgoszczy konto nr:5.09011-400522.7-136-11-111.0 następujących kwot: 20.000zł za pojedyńczy zestaw dla C-64, 25.000zł za zestaw dla AMIGI. Kwoty te obejmują koszt dyskietki, koszty kopiowania, opakowania i przesyłki pocztowej. Blankiety wpłat powinny być CZYTELNIE wypełnione i zawierać: imię i nazwisko, dokładny adres zamawiającego, skrót „PDP-64"(jeśli zamawiamy zestaw dla C-64) lub „PDP-A"(zestaw dla Amigi). W prenumeracie zestawy kosztują: PDP-61 - 18.000zł (12 numerów 216tys zł), PDP-A - 22.000zł (12 numerów 264tys zł). Prenumeratę można zawrzeć w dowolnym terminie na okres od 3 do 12 miesięcy (do końca roku kalendarzowego). Prenumerata może obejmować miesiące od początku roku - tzn. zamawiając całoroczną prenumeratę np. w kwietniu, w pierwszej przesyłce otrzymacie wszystkie poprzednie zestawy (tj. styczeń, luty i marzec).

Zmagania z Copperem cz.3

Tworzymy CopperList

W poprzednich artykułach naszego cyklu zapoznaliśmy się z rejestrami potrzebnymi do tworzenia obrazu. Teraz przyszła kolej na napisanie programu dla procesora wizji (Copper'a) czyli tzw. CopperList. Copper posiada trzy rodzaje rozkazów - są to: MOVE, WAIT i SKIP. Nazwy są tylko umowne, gdyż CopperList tworzymy z liczb odpowiadających tym rozkazom. Każdy rozkaz Copper'a składa się z dwóch słów (32 bity).

Instrukcja WAIT (czekaj)

Copper zatrzyma się na tym rozkazie do czasu gdy liczniki ramki x i y osiągną wartości podane jako parametry dla tego rozkazu. Jest to tzw. oczekiwanie na „raster".

Pierwsze słowo instrukcji WAIT:

bit 0	-zawsze ustawiony na 1;
bity 15-8	-pozycja w pionie, na którą czeka Copper (Vertical Position);
bity 7-1	-pozycja w poziomie (Horizontal Position).

Drugie słowo instrukcji WAIT

bit 0	-zawsze wyzerowany;
bit 15	-bit zakończenia pracy przez blittera (ustawiony na 1);
bity 14-8	-bity maski dla VP - które bity mają być porównywane;
bity 7-1	Bity maski dla HP - które bity mają być porównywane.

Przykłady wykorzystania instrukcji WAIT

```
dc.w $8435,$fffe    ;oczekiwanie na linię $84
                     i punkt $1b w tej linii;

dc.w $c401,$ff00    ;oczekiwanie na linię $c4
                    z pominięciem czekania na
                    pozycję poziomą (HP);

dc.w $ffff,$fffe    ;oczekiwanie na ostatni punkt
                    ekranu (kończy wykonywanie
                    CopperList'u);

dc.w $ffdf,$fffe    ;koniec górnej połówki ekranu
                    (dla PAL) każde odwołanie
                    do instrukcji WAIT będzie
                    dotyczyło dolnych linii ekranu.
```

Instrukcja MOVE (przenieś)

Pierwsze słowo jest liczbą, do której dodawany jest adres Custom Chip ($dff000), a następnie pod obliczony adres jest wpisywane drugie słowo tego rozkazu.

Pierwsze słowo instrukcji MOVE :

bit 0	-zawsze wyzerowany;
bity 8-1	-ofset względem Custom Chip
bity 15-9	-nieużywane (muszą być ustawione na 0)

Drugie słowo instrukcji MOVE :

bity 150	-dana, która będzie przeniesiona pod adres przeznaczenia.

Przykłady instrukcji MOVE

```
dc.w $0180,$0f00    ;ustawienie koloru nr 00
                    ($dff180 - tło) na$f00 (czerwony);

dc.w $00e0,$0006    ;ustawienie adresu pierwszego
                    bitplane'u;

dc.w $00e2,$2800    ;(BPL1PT) na adres $62800.
```

Instrukcja SKIP (pomiń)

Rozkaz ten nakazuje Copper'owi pominąć następny rozkaz CopperList'u, jeżeli zawartość liczników pozycji pionowej i poziomej (VP i HP) jest większa lub równa pozycji podanej w instrukcji SKIP.

Pierwsze słowo instrukcji SKIP :

bit 0	-zawsze ustawiony na 1;
bity 15-8	-pozycja w pionie, na którą czeka Copper (Vertical Position);
bity 7-1	-pozycja w poziomie (Horizontal Position);

Drugie słowo instrukcji SKIP :

bit 0	-zawsze ustawiony na 1;
bit 15	-bit zakończenia pracy przez blittera (ustawiany na 1);
bity 14-8	-bity maski dla VP - które bity mają być porównywane;
bity 7-1	-bity maski dla HP - które bity mają być porównywane.

Przykłady instrukcji SKIP

```
dc.w $a601,$ff01    ;pomiń następną instrukcję
                    jeżeli licznik ramki(VP) jest
                    większy lub równy $a6.
```

Na zakończenie przykładowy CopperList. Co on wykonuje proponuję sprawdzić czytelnikom samodzielnie.

```
custom:     = $dff000
cop1lc:     =$80
copjmp1:    =$88
            lea     custom,a6
            move.l  #Copper,cop1lc(a6)
            clr.w   copjmp1(a6)
            moveq   #0,d0
Copper:
            dc.w    $0100,$0200
            dc.w    $0180,$0000
            dc.w    $8007,$fffe
            dc.w    $0180,$0f00
            dc.w    $8107,$fffe
            dc.w    $0180,$0000
            dc.w    $ffff,$fffe
END
```

W następnym odcinku Zmagań z Copper'em postaram się opisać programowanie CopperList'u czyli np. tworzenie pętli.

Marcin „Duddie" Dudar

A może byśmy coś poudawali... (cytat z Mrożka) na Amidze

Nie trzeba nikogo przekonywać, że Amiga w swojej klasie cenowej jest najlepszym komputerem na świecie. Niestety, standardem krajowym stały się tajwańskie klony IBM i jeśli nie zamierzasz używać Amigi wyłącznie do zabawy - aby nie wylecieć z obiegu - musisz mieć z Wielkim Niebieskim [------] choć trochę wspólnego.

Mam nadzieję, że po przeczytaniu tego wstępu nie popędzisz na najbliższą giełdę aby pozbyć się Amigi. Na Twoim wspaniałym komputerze również można poudawać IBMa i to na kilka sposobów. Omówię niektóre z nich w ujęciu chronologicznym.

Dawno temu posiadacze A500 patrzyli zazdrosnym okiem w stronę użytkowników A1000, którzy mogli emulować IBM XT za pomocą dodatkowego urządzenia o nazwie SIDECAR. Było to pudełko zawierające płytkę z emulacją 4.77 MHz IBM oraz stację dysków 5.25 cala. Po podłączeniu do A1000 i uruchomieniu dyskietki z programem emulującym A1000 przekształcała się w IBM XT. Jednocześnie na monitorze Amigi emulowane były dwa tryby pracy: monochromatyczny tekstowy (MDA - nie należy mylić z Herculesem, który ma symbol MGA) oraz graficzny kolorowy (CGA). Jakkolwiek sama emulacja IBM została wykonana poprawnie - obsługa ekranu zawierała szereg bugów, w związku z czym IBM udawany na A1000 pracował ok. 2-krotnie wolniej niż oryginalny XT.

Kolejnym problemem była pamięć. A1000 miała jej jedynie 256 kB. Konstruktorzy Sidecara poszli po najmniejszej linii oporu wpakowując do emulatora 512 kB pamięci IBM'owskiej. Wpłynęło to niewątpliwie na cenę, która początkowo wynosiła ok. 650$. Urządzenie nie cieszyło się jednak zbyt dużym popytem (podobnie jak i sama A1000) toteż najpierw obniżono cenę, a później firma Rossmoeller odkryła, że w prosty sposób (krosując kilka przewodów) da się połączyć Sidecara do A500. Niestety, w międzyczasie Commodore zaprzestała produkcji urządzenia. W tej chwili można je nabyć wyłącznie z ogłoszenia bądź na wyprzedażach, ale przy cenie oscylującej w granicach 200$ jest to rozwiązanie w miarę niedrogie.

ZALETY: niska cena, możliwość wykorzystania niektórych cech Amigi przy pracy „pod IBM'em" (w palecie barw IBM można wykorzystać dowolny kolor z palety Amigi, przekopiowywać zbiory danych z Amigi na IBM'a i odwrotnie), możliwość podpięcia do A500.

WADY: wolna praca, zwłaszcza przy obsłudze ekranu, duża ilość zajmowanego miejsca, mała ilość pamięci w trybie IBM'a, długie instalowanie (program emulujący wczytuje się ok.4 minut).

Oczekiwany przez wielu posiadaczy A2000 BRIDGEBOARD XT okazał się jedynie nieco zmodyfikowanym SIDECARem. Chociaż dodano w nim kilka nowych możliwości (np. obsługa dysku twardego, tworzenia IBM'owskiego RAM-dysku, czy wirtualnej emulacji twardego dysku), tym niemniej odziedziczył on wszystkie wady swego poprzednika. Jednocześnie producent wyszedł z założenia, że posiadacze A2000 to piekielnie bogaci ludzie, w związku z czym początkowa cena karty była wyższa niż samego komputera. Z czasem na szczęście spadła i obecnie można ją nabyć od 420$.

I znów firmy elektroniczne wykryły kilka interesujących możliwości: m.in. to, że wymieniając w karcie procesor INTEL na NEC-V20 można uzyskać taktowanie 8 MHz. Wypuszczono także kartę rozszerzającą pamięć w trybie IBM'a o brakujące do szczęścia 128 kB.

Najdalej poszła jednak amerykańska firma ETHEL wypuszczając rewelacyjną ponoć (nie miałem okazji sprawdzić) kartę emulującą IBM 386 na emulowanym na Amidze XT. Rozumiesz to ostatnie zdanie? Nie? Nie szkodzi - to jest i tak bez pojęcia.

ZALETY: możliwość schowania w obudowę A2000, możliwość usprawnienia.

WADY: prócz wszystkich wymienionych przy SIDECAR'ze - wysoka cena, oraz to, że montaż wymaga zdjęcia obudowy co może być równoznaczne z utratą gwarancji na komputer.

W grudniu 1988 ukazał się szumnie zapowiadany BRIDGEBOARD AT. I cóż się okazało. Za cenę 1450$ otrzymywałeś BRIDGEBOARD XT, tyle, że pracujący z taktowaniem 8 MHz (w tym czasie już istniały BIOSy pozwalające pracować zwykłym XT'ciakom z taktowaniem 10 MHz), mający ponadto pełne 640 kB pamięci i stację 1.2 MB, a także zasilany bateryjnie zegar (ulubione siedlisko wirusków). Wprawdzie i tu cena z czasem spadła, ale z tego co wiem jedynie 4 posiadaczy A2000 w kraju dało się nabrać na ten „bajer".

W tym momencie zacząłem przypuszczać, że nie doczekamy się „ludzkiej" emulacji IBM'a na Amidze, co było dziwne - gdyż zawsze da się emulować gorszy komputer na lepszym. Szerokiej rzeszy posiadaczy Amigi 500 pozostawał TRANSFORMER - niewydarzona próba dostosowania programu emulującego z SIDECARa do 500-tki. TRANSFORMER rzekomo był emulatorem AT. Jednak przy obsłudze ekranu pobito wszelkie rekordy i wydawać się mogło, że udawany IBM pracuje z taktowaniem 0.0000001 MHz. Ponadto można było wgrać programy, których wielkość nie przekraczała 371 kB, zaś Amigowskich rozszerzeń pamięci program uprzejmie nie raczył zauważać. Ponadto występowały olbrzymie problemy z grafiką. W momencie gdy po wielu próbach doszedłem do tego, że program uznał za swoje jedynie trzy rozkazy grafiki CGA (LINE,CIRCLE i PSET) - zmieniłem A500 na A2000 i tu okazało się, że Transformer nie „lubi się" z 1MB Chip RAM.

Na szczęście koniec 1990 roku przyniósł dwa nowe (na razie dosyć drogie) rozwiązania.

Firma VORTEX wypuściła emulator AT o nazwie ATonce (at once - po angielsku znaczy - nareszcie) w cenie 310$. Są to dwie malutkie płytki . Jedna z nich zawiera hardware IBM'a, druga zaś to zmodyfikowany układ GARY. Instalacja hardware, nie dość, że może spowodować utratę gwarancji - jest wyjątkowo skomplikowana. Poza podmianą GARYego (oryginalny chip

możesz sobie zachować) - co jest rzeczą średnio trudną - musisz pamiętać o tym , że technologia SMT, w której wykonano właściwą płytę PC, nie zapewnia ani procenta zabezpieczenia przed przebiciami elektrostatycznymi. Wystarczy, że jesteś podekscytowany zakupem nowego emulatora lub masz na sobie koszulę non-iron, a już po hardwarze. Jeśli jednak (być może z niewielką pomocą przyjaciół fachowców) przebrniesz przez ten etap -pozostaje Ci już tylko zdobyć kopię MSDOS'a (nie dołączona do emulatora!), a dalej... sama przyjemność.

Emulator zapewnia pełny wielodostęp pod MSDOS'em (ten na BRIDGEBOARD'zie był nieco oszukany). Można pracować w 3 trybach graficznych: CGA, Hercules(zwykły i Toshiba 3100) oraz VGA(którą jednak trzeba instalować oddzielnym programem). UWAGA: im wyższa rozdzielczość i im więcej kolorów tym wolniejsza jest obsługa ekranu. Dysk A> jest emulowany na dysku Df0:.Jeśli masz dodatkową stację możesz na niej emulować dysk B>.Nie ma możliwości emulacji HD. Pod względem zgodności z oryginałem ATonce jest jednak gorszy od swego opisanego niżej rywala, jednak tylko w jednej dziedzinie. Z niezrozumiałych powodów nie chodzi żadna(!) gra z PC. Posiadacze A2000, którzy chcą użytkować ATonce muszą dokupić dodatkowy adaptor w cenie 25$. Ale i tak jest to rozwiązanie trzykrotnie tańsze niż BRIDGEBOARD AT. W Anglii powstał nawet ostatnio klub użytkowników ATonce.

Na koniec jednak zostawiłem rodzynek, jakim jest zapowiadany od dawna emulator firmy KCS, który wreszcie się pojawił. Kosztuje co prawda aż 400$ (w tym IBM'owski software (MS DOS 4.1) wart zdaniem firmy 150$, lecz bez niego jakoś nie chcą sprzedawać), ale za tą cenę masz: dziecinnie łatwą instalację, nie naruszającą gwarancji, dodatkowe 1 MB pamięci na Amidze (w tym 512 kB bufora dyskowego co bardzo przyspiesza pracę stacji), ponadto modulator TV, trzy tryby pracy MDA, CGA i Hercules (bez spowolnienia pracy przy obsłudze ekranu). W trybie pracy IBM emulowana jest praca drukarki, stacji dysków i twardego dysku na odpowiednich urządzeniach A500. Masz do dyspozycji 704(!!!) kB pamięci IBM'a oraz 64 kB EMS. Firma twierdzi, że chodzą wszystkie programy z IBM'a, gwarantując jednocześnie, że każdy „niechodliwy" program zostanie przez nich bezpłatnie przystosowany.

Miejmy nadzieję, że nie jest to ostatnie słowo w dziedzinie emulatorów IBM'a na Amidze.

A co robi konkurencja czyli „druga strona"? Wypuściła „emulator" Amigi na IBM'ie. Miałem go w ręku wyłącznie przez chwilę, bowiem zdyskwalifikował się w moich oczach faktem, że przenosi wyłącznie 16 kolorów (nawet na SVGA - spróbujcie zagrać np. w bajecznie barwną, choć głupawą Pionieer Plague pod tym emulatorem - grafika będzie gorsza niż na ZX81), a już zupełnie „dobił" go moment, w którym najpierw uruchomiłem „emulator" IBM'owski, a gdy zgłosiło się coś co miało przypominać Workbench spróbowałem wczytać Transformera. Nie zdradzę efektu. To każdy MUSI zobaczyć SAM!

Marek Pampuch

SCULPT 4D cz. 1

Fantastyczne możliwości graficzne Amigi (HAM, Hi-Res w 16 kolorach) były często wykorzystywane w różnej formie przez programy graficzne (Photon Paint), digitizery czy też przez programy Ray Tracing.

Ray Tracing czyli śledzenie promienia to kopiowanie przestrzeni trójwymiarowej do płaskiego rysunku, ale wraz z oświetleniem płaszczyzn, odbiciami, cieniami itp. Metoda ta polega na śledzeniu promienia światła i sprawdzeniu w jakim miejscu przebija on płaszczyznę ekranu, po czym chrominancja i luminancja tego promienia jest zamieniana na właściwy kolor tego punktu na ekranie. W tym celu wykorzystywane są skomplikowane wzory matematyczne i fizyczne a sama metoda wymaga użycia zaawansowanych technik programowania. Jednak już kilka lat temu powstały pierwsze programy tego typu, a obecnie najlepszymi z nich są Sculpt 4D i 3D Professional. My zajmiemy się tym pierwszym ponieważ daje się on uruchomić nawet na 0.5MB RAM'u. Ten drugi zaczyna normalną pracę od 2MB i z tego powodu nie jest dostępny dla zbyt wielu użytkowników.

Autorem Sculpt'a 4D jest Eric Graham. Sculpt 4D jest bardzo rozbudowanym programem, a co najważniejsze, bardzo wygodnym i prostym w obsłudze. Najważniejsze funkcje są osiągalne z klawiatury, a samo projektowanie przestrzeni trójwymiarowej jest proste i można je opanować w bardzo krótkim czasie.

Sculpt 4D ma swoją podręczną bibliotekę podstawowych figur geometrycznych o mniejszym lub większym stopniu skomplikowania. Pozwala on na generację różnego rodzaju źródeł światła, tworzenie płaszczyzn metalicznych, błyszczących, matowych, lśniących, lustrzanych i innych. Sam proces rysowania może być zmieniany w zależności od tego jakiej jakości obraz chcemy uzyskać. Położenie obserwatora oraz jego nachylenie do płaszczyzny może być dowolnie zmieniane a sam rysunek możemy uzyskać w wielkości jaka jest nam potrzebna.

Program Sculpt 4D jest doskonałym narzędziem zwłaszcza, że można na nim tworzyć animacje obrazów (do wykorzystania przy produkcji np. własnych czołówek video). Program ten ma tylko jedną poważną wadę - nie należy do najszybszych. Wykreowanie rysunku w największym forrmacie, ze szczegółami, gdy będziemy mieli kilkaset płaszczyzn, a zwłaszcza lustrzanych, może potrwać i kilka dni. Nie należy się jednak zrażać - efekt końcowy przerośnie nasze oczekiwania!

Tych, których zainteresował ten program zachęcamy do lektury kolejnych odcinków naszej małej instrukcji do Sculpt'a 4D.

Marcin „Duddie" Dudar

Kącik początkującego kodera

Assembler zaliczany jest do kategorii języków symbolicznych. Umożliwia on programiście zastąpienie liczbowych kodów rozkazów i danych języka maszynowego łatwiejszymi do zapamiętania nazwami symbolicznymi. Czym natomiast jest język maszynowy? Jest to ciąg liczb tworzących program, które są rozpoznawane przez mikroprocesor, a następnie wykonywane. Jako, że pisanie programów jako ciągów liczbowych byłoby bardzo niewygodne, stworzono odpowiedni język - assembler. Wszystko to po to, aby ułatwić pracę programistom (właściwie koderom, ponieważ programowanie w assemblerze nazywa się kodowaniem). Proces translacji, czyli tłumaczenia programu napisanego w języku assemblera na język maszynowy nazywa się assemblacją. Obecnie dostępne są różne wersje „Assemblera Seka" na którym polecam rozpoczęcie nauki i późniejsze kodowanie.

Pisząc program w języku assemblera tworzymy tzw. program źródłowy (ang. source), z którego po procesie assemblacji powstaje kod wynikowy (ang: object code), który może być wykonywany przez mikroprocesor.

MOTOROLA 68000 jest mikroprocesorem 16-bitowym z 24-bitową szyną adresową. Dane, które potrafi interpretować i wykonywać na nich działania (z wyjątkiem mnożenia i dzielenia) są 32-bitowe. Ktoś może zadać pytanie: „To dużo czy mało?". Oznacza to możliwość operacji na liczbach z zakresu od 0 do (2^{32})-1, czyli od 0 do 4294967295. Często mokroprocesory ocenia się pod względem ilości rejestrów dostępnych użytkownikowi. M68000 posiada 8 rejestrów danych ponumerowanych kolejno: d0,d1,d2,...,d7 używanych jako ośmio-, szesnasto- lub trzydziestodwu bitowe rejestry, służące głównie do wykonywania działań arytmetycznych, chwilowego przechowywania danych, indeksowania pamięci. Poza rejestrami danych istnieje 8 rejestrów adresowych, oznaczonych a0,a1,a2,...,a7. Rejestry te używane są jako szesnasto- lub trzydziestodwu bitowe i służą do czasowego przechowywania adresów lub oznaczają miejsca, skąd mają być czerpane dane. Powyższe odnosi się do rejestrów a0-a6. Inne zadania posiada natomiast rejestr a7. Służy on jako tzw. wskaźnik stosu.

Czym jest stos? Jest to fragment pamięci przeznaczony do czasowego przechowywania danych, przy czym tylko ostatnio wprowadzona dana jest dostępna. Stos może przypominać stertę książek poukładanych jedna na drugiej. Zauważmy, że tylko ostatnio położona książka jest widoczna i tylko ją możemy zdjąć, następną zaś dopiero po zdjęciu tej pierwszej. Taka zasada nosi nazwę LIFO (ang: Last In First Out) czyli ostatni wszedł, pierwszy wyszedł. Stos wykorzystywany jest zarówno przez użytkownika jak i sam procesor. Użytkownik używa go do chwilowego przechowywania danych, w szczególności zawartości rejestrów. Natomiast mikroprocesor odkłada na stos licznik programu (ang: PC - czyli adres aktualnie wykonywanej instrukcji) przy skoku do podprogramu lub procedury obsługi przerwania, aby po napotkaniu rozkazu powrotu zdjąć ten adres (ze stosu oczywiście) do licznika programu i w ten sposób wrócić z podprogramu do programu głównego.

Ale wróćmy może do rejestrów. Jak już wspomniałem rejestr adresowy a7 jest wskaźnikiem stosu. Wskazuje adres pod który ostatnio wprowadzono daną. Tak więc przy zdejmowaniu danej należy dodać do rejestru a7 długość danej (1,2 lub 4 bajty w zależności od tego czy przesyłamy bajt, słowo czy długie słowo), lub odjąć tę długość przy odkładaniu na stos. Ufff... Trochę to skomplikowane, jednak zapewniam, iż w praktyce okazuje się to zupełnie proste.

Następnym, ważnym rejestrem jest rejestr systemowy zwany również rejestrem statusu lub stanu mikroprocesora, składający się z dwóch bajtów:

bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
	T	.	S	.	.	*	*	*	.	.	.	X	N	Z	V	C
	Bajt systemowy								Bajt użytkownika							

Oznaczenia:
BAJT SYSTEMOWY:
T - (ang: trace bit) bit śledzenia - krokowe wykonywanie programu
S - (ang: supervisor bit) bit kontroli pracy procesora
* - maska przerwań.

Nie będę szerzej opisywał tego bajtu, gdyż jego znajomość nie jest konieczna dla początkującego kodera. Znacznie bardziej znaczący jest: BAJT UŻYTKOWNIKA.

Bajt ten zawiera zestaw bitów zwanych kodami warunków. Charakteryzują one wynik ostatnio wykonanej instrukcji. Dostępne są następujące bity:

1. Bit 0 (ang: carry) - bit przeniesienia. Ustawiany lub zerowany jest w operacjach arytmetycznych i logicznych. Używany również w przesunięciach i obrotach bitów.
2. Bit 1 (ang: overflow) - bit przepełnienia. Używany, gdy wynik operacji nie może być prawidłowo wyrażony. Rzadko sprawdzany przez koderów.
3. Bit 2 (ang: zero) - bit zera. Ustawiany zawsze, gdy wynikiem ostatniej operacji było zero, w przeciwnym razie zawsze kasowany.
4. Bit 3 (ang: negative) - bit ujemności. Ustawiany, gdy wynikiem ostatniej operacji jest liczba ujemna, a kasowany jeśli dodatnia.
5. Bit 4 (ang: extend) - bit rozszerzenia. Bit ten jest kopią bitu C, ale nie jest zmieniany przez każdą instrukcję, która zmienia C.

Wszystkie informacje znajdujące się w pamięci komputera są liczbami. Grafika, muzyka czy nawet teksty również zapisywane są za pomocą liczb odpowiednio kodowanych (w przypadku tekstów mamy do czynienia z szeroko stosowanym kodem ASCII). Dlatego też bardzo ważną będzie dla nas umiejętność sprawnego posługiwania się liczbami.

W życiu codziennym mamy do czynienia z tzw. dziesiętnym (ang: decimal) zapisem liczb. Został on przyjęty bardzo dawno i przyzwyczailiśmy się do niego - widząc liczbę np. 156 nie zastanawiamy się „skąd TO się wzięło". My jednak zatrzymamy się przy tym problemie na dłużej, aby później łatwiej zrozumieć system zapisu jaki funkcjonuje w komputerach.

$$156 = 100 + 50 + 6 = \\ = 1*100 + 5*10 + 6*1 = \\ = 1*10^2 + 5*10^1 + 6*1^0 =$$

Jak widać liczba dziesiętna jest to suma iloczynów cyfr (w systemie dziesiątkowym mamy 10 cyfr od 0 do 9) przez kolejne potęgi liczby 10. Myślę, iż jest to zrozumiałe. Niestety mikroprocesor rozpoznaje i wykonuje działania na liczbach zapisywanych w systemie binarnym lub inaczej dwójkowym. System taki został przyjęty ze względu na to, że elektryczne układy komputera mogą znajdować się w dwóch stanach:

1 - włączony (płynie prąd)

0 - wyłączony (prąd nie płynie)

Analogicznie jak w systemie dziesiętnym tak i w dwójkowym liczba binarna jest sumą iloczynów cyfr (tu mamy dwie cyfry: 0 i 1) przez kolejne potęgi liczby 2.

$$156=1*128+0*64+0*32+1*16+1*8+1*4+0*2+0*1= \\ =1*2^7+0*2^6+0*2^5+1*2^4+1*2^3+1*2^2+0*2^1+0*2^0= \\ =\%10011100$$

W ten właśnie sposób wygląda decymalne 156 zapisane w systemie dwójkowym (binarnym). Znak „%" przed tą liczbą jest symbolem stosowanym przez wszelkiego rodzaju monitory pamięci oraz assemblery do oznaczania liczb dwójkowych.

Używanie zapisu binarnego byłoby dla nas bardzo niewygodne, wręcz absurdalne. Dlatego też, z powodzeniem przyjął się inny system : szesnastkowy lub inaczej heksadecymalny. Taka liczba (zazwyczaj oznaczana znakiem $) jest sumą iloczynów cyfr przez potęgi liczby 16. W systemie szesnastkowym mamy 16 cyfr oznaczanych : 0,1,2,..., 8,9,A,B,C,D,E,F. Jak widzimy $A = dec10, a $F = dec15. Liczby heksadecymalne są wygodne w użyciu, ze względu na to, że cyfra hex reprezentuje połówkę bajtu. Zobaczmy jak to wygląda na przykładzie naszej 156:

%10011100 =	1001	1100
dec:	9	12
hex:	$9	$C

Tak więc: dec156 = %10011100 = $9C.

Najmniejszą jednostką informacji jaką może rozpoznać i przetworzyć procesor jest bit, czyli cyfra binarna (0 lub 1). Większą jednostką jest bajt. Składa się on z ośmiu bitów i może przyjmować wartości od 0 do 255 (od $00 do $FF). Dwa bajty tworzą słowo (ang: word), natomiast dwa słowa długie słowo (ang: long word). To ostatnie składa się z 32 bitów.

Myślę, że po tej nudnej gadaninie, a w zasadzie pisaninie nadszedł czas na poznanie pierwszych rozkazów mikroprocesora.

MOVE.x <źródło>,<przeznaczenie>

Jak sama nazwa wskazuje, instrukcja służy do przesyłania danej z operandu źródła do miejsca przeznaczenia. Indeks „x" za instrukcją mówi nam jakiej wielkości daną mamy przesyłać. Możliwe są trzy przypadki:

b - przesyłamy bajt czyli 8 bitów
w - przesyłamy słowo czyli 16 bitów
l - przesyłamy długie słowo tzn. 32 bity

ADD.x <źródło>,<pzeznaczenie>

Instrukcja ta dodaje operand źródłowy do miejsca przeznaczenia ; wynik zostawia w miejscu przeznaczenia.

SUB.x <źródło>,<przeznaczenie>

Instrukcja SUB odejmuje operand źródłowy od operandu przeznaczenia i tam też zostawia wynik. Spójrzmy teraz na mały przykład:

```
move.b  #$20,d0 ; przesyłamy liczbę $20 do
                  rejestru danych d0
move.b  #$0a,d1 ; przesyłamy liczbę $0a do
                  d1
add.b   #$07,d0 ; dodajemy liczbę $07 do
                  rejestru d0; teraz w d0 mamy
                  wartość $27
sub.b   d1,d0   ; odejmujemy zawartość rejestru
                  d1 (czyli $0a) od zawartości
                  rejestru d0 (czyli $27)
                  i otrzymujemy wynik
                  $27-$0a=39-10=29=$1d
                  który umieszczony jest w
                  rejestrze d0
```

Mechanizm wykonywania działań arytmetycznych jest bardzo prosty i powinien być zrozumiały. Na przykładzie instrukcji MOVE przedstawię teraz tryby adresowania mikroprocesora 68000, które określają skąd mają być czerpane dane do wykonywanej instrukcji.

1. Tryb adresowania bezpośredniego rejestru danych

Format: Dn n - numer rejestru (0-7)

Przykład: move.b d0,d2

Zawartość rejestru d0 w rozmiarze bajtu zostanie przesłana do rejestru d2, przy czym zmienione zostanie tylko 8 najmłodszych bitów.

Przykład: move.l d3,d7

Rejestr d3 zostanie przesłany do d7 - długie słowo, więc zostaną zmienione wszystkie 32 bity.

2. Tryb adresowania bezpośredniego rejestru adresowego
Format: An n - numer rejestru (0 do 7)
Przykład: move.l a4,d3
Rejestr adresowy (poprawnie powinno się pisać „zawartość rejestru a4", ale w slangu używa się powyższej formy) zostanie przesłany do d3 i oba będą miały taką samą zawartość. W tym trybie przesyłanie ograniczone jest do słowa i długiego słowa, natomiast nie wolno przesyłać bajtu.

3. Tryb adresowania pośredniego rejestrem adresowym
Format: (An) n - numer rejestru
Przykład: move.l (a0),d1
Do d1 zostanie przesłane długie słowo spod adresu pamięci, który wskazuje rejestr a0.

4. Tryb adresowania pośredniego rejestrem adresowym z postinkrementacją
Format: (An)+ n - numer rejestru
Przykład: move.b (a4)+,d1
Do d1 zostanie przesłane słowo spod miejsca w pamięci które wskazuje a4, a następnie a4 zostanie powiększony o 2.
Gdybyśmy przesyłali bajt, a4 zostałby powiększony o 1, natomiast w przypadku długiego słowa o 4.
Tryb ten najczęściej używany jest do przenoszenia dużych obszarów pamięci.

5. Tryb adresowania pośredniego rejestrem adresowym z predekrementacją
Format: -(An) n - numer rejestru
Przykład: move.l -(a0),d3
Rejestr a0 zostanie pomniejszony o 4, następnie spod adresu wskazanego przez a0 zostanie przesłane długie słowo do d3.
W przypadku przesyłania bajtu a0 zostanie zmniejszony o 1, słowa o 2.

6. Tryb adresowania pośredniego rejestrem adresowym z przesunięciem
Format: x(An) n - numer rejestru
x - 16-bitowe przesunięcie
Przykład: move.l $08(a0),d6
Do d6 zostanie przesłane długie słowo spod adresu, który jest sumą zawartości rejestru a0 i przesunięcia (w tym wypadku równego 8). Zawartość a0 zostaje niezmieniona. W tym trybie przesunięcie może mieć wielkość do 16 bitów.

7. Tryb adresowania pośredniego rejestrem adresowym z indeksem i przesunięciem
Format: x(An,Dm.w) n,m - numery odpowiednich rejestrów (0 do 7)
x(An,Dm.l) x - 8 bitowe przesunięcie
x(An,Am.w)
x(An,Am.l)
Przykład: move.l $23(a0,d1.w),d7
Do d7 zostanie załadowane długie słowo spod adresu, który jest sumą zawartości rejestru a0, rejestru d1 (w wymiarze słowa - czyli dodane będzie tylko słowo z d1), oraz przesunięcia czyli $23. Zawartości a0 i d1 pozostają niezmienione. W tym trybie adresowania przesunięcie jest 8-bitowe więc może przybierać wartości od 0 do 255.

8. Tryb adresowania absolutnego krótkiego
Format: adres.w adres - 16-bitowy adres
Przykład: move.l 4.w,d0
Do d0 zostanie przesłane długie słowo spod adresu 4 (czyli z komórek 4,5,6 i 7). Tego trybu używamy, gdy chcemy pobierać dane spod adresów mieszczących się na 16 bitach.

9. Tryb adresowania absolutnego długiego
Format: adres adres - 24-bitowy adres
Przykład: move.b $50000,d1
Do d1 zostanie przesłany bajt spod adresu $50000.

10. Tryb adresowania licznikiem programu z przesunięciem
Format: x(pc) x - 16-bitowe przesunięcie
Przykład: move.l $124(pc),d1
Do d1 zostanie załadowane długie słowo spod adresu, który jest sumą PC (czyli adresu tej instrukcji) oraz przesunięcia czyli- w naszym przykładzie -$124.
Przesunięcie jest wartością mieszczącą się w 16 bitach.

11. Tryb adresowania licznikiem programu z indeksem oraz przesunięciem
Format: x(pc,Dn.w) x - 8-bitowe przesunięcie
x(pc,Dn.l) n - numer rejestru
x(pc,An.w)
x(pc,An.l)
Przykład: move.b $18(pc,d4.l),d0
Spod adresu który jest sumą PC, 8-bitowego przesunięcia oraz długiego słowa zawartego w rejestrze d4 będzie przesłany bajt do rejestru danych d0.

12. Tryb adresowania natychmiastowy
#liczba liczba - wartość mająca być przesłana
Przykład: move.l #$dff000,d1
Wartość $dff000 zostanie umieszczona w rejestrze danych d1.

13. Tryb adresowania rejestru statusowego
Ten tryb służy do programowych zmian rejestru statusowego.

Uff... to by były wszystkie tryby adresowania.
Jeżeli chcecie zapoznać się z następnymi instrukcjami mikroprocesora Motorola MC68000 i z przykładowymi programami, napisanymi w asemblerze, to czytajcie kolejne artykuły z tego cyklu w „64+4"!

Krzysztof „K.K." Kobus